ISSN 0998-9250 Juin 1991 / N° 14

Avec cahier Dr Dobb's Journal

DEBUG magazine

SYSTEMES MULTI-TACHES MULTI-UTILISATEURS

UN NOYAU MULTI-TACHES EN C++

DDL : ECHANGE DE DONNEES SOUS WINDOWS

Choisir son réseau :
Appel d'offres

Tasse de Thé :
Dessin 3 D

Initiation :
Récursivité

Page de John C. Dvorak

M 3032 - 13 - 37,00 F

Belgique : 258 FB - Suisse : 11,5 frs

ISSN 0998-9250 - Hors-Série N° 1

DEBUG magazine

23 PROGRAMMES DE LECTEURS

JEUX
MAINTENANCE
GRAPHIQU
OUTILS
TE

ACTUELLEMENT EN KIOSQUE HORS - SERIE N° 1

Belgique : 423 FB - Suisse : 18, 30 FS

EDITO

DOMAINE PUBLIC, PROPRIETE PRIVEE ?

Un jour, un juriste s'est penché sur le charabia informatique.
Car, par le temps de vaches maigres, même le jargon informatique peut donner lieu à des tentatives de récupération. Imaginez le résultat ...

En substance, rien de nouveau. Notre industrie est infectée par d' interminables procès qui ne font rien avancer et qui n'enrichissent que les avocats. Pourquoi alors en parler? Car cette fois-ci, l'opinion de serviteur de Temide concerne un mot, le mot qui nous intéresse bigrement : « DEBUG ». La manoeuvre est claire : quelqu'un veut se l'approprier.

Selon l'homme de loi, le mot « DEBUG » peut être une marque privée, une enseigne commerciale, il n'est descriptif d'aucun produit, ni service désigné. « Les termes « DEBUG », « BUG », « DEBUGGING » sont des termes anglais usuels d'informatique figurant dans les notices et modes d'emploi de langue anglaise mais que les néologismes français « debugage », « debuguer » ne sont pas utilisés couramment en France ». En plus, toujours selon la même opinion, « debugging » n'est pas le terme descriptif du produit ou service désigné. Le mot « DEBUG » serait absent dans le langage informatique français.

En conséquence, on peut se le réserver, privatiser, vendre.
Depuis que nous connaissons ce métier, le mot Debug nous est connu. Non seulement en français mais aussi en anglais, italien, allemand, russe et chinois. Dieu seul sait qui l'a inventé. Notre métier crée ses propres termes que seul un initié peut comprendre.
Que veut dire « bootstrap », par exemple, pour un anglais ?
Rien pour le commun de mortels, tout pour certains.

Idem Debug, ou débogage. L'obscure confidentialité de notre langage doit donner à certains le droit de s'approprier un mot appartenant au domaine public, pour le vendre? Peut-on vendre les mots d'une langue? Nous pensons que non. Nous estimons que le patrimoine de notre industrie, y compris son langage, appartient à tous, ne peut pas être privatisé. Comme les logiciels ou algorithmes du domaine public. Comme les langages de programmation. Et nous pensons que cela vaut la peine d'être défendu.

Nous voudrions connaître votre opinion à ce sujet.
Écrivez-nous, vos opinions vont être publiées.

Artur G Piotrowski

PARLONS - EN

Joignez le dialogue! Écrivez à notre adresse :

DEBUG MAGAZINE
14 rue de l'Atlas
75019 PARIS
ou sur :
CompuServe 71021,1666
Calvacom SP59

Tous les lettres deviennent la propriété de **DEBUG MAGAZINE** et ne peuvent être retournées. Nous nous réservons le droit d'éditer le texte, pour les raisons d'espace ou de la clarté.

Artur G Piotrowski
rédacteur en chef

HORS PAIR ET HORS SÉRIE

☛ ***« Je suis actuellement étudiant en informatique (j'ai fait un DUT puis une année de spécialisation Interfaces Homme-Machine et Réseaux), et lecteur non assidu de votre magazine. Je viens d'acheter le Hors Série n° 1 et suis au regret de vous signaler qu'il n'y pas dans ce numéro cinquante huit francs de programmes! Le montant du capital de votre SARL me laisse cependant penser que vous vous battez comme de beaux diables pour faire ce magazine. Aussi, j'en profite pour vous tirer un coup de chapeau. J'admire ceux qui se battent. Je voudrais par cette lettre vous signifier mon intérêt pour votre dossier collaborateur. Ce n'est pas la rémunération qui m'attire, mais la possibilité d'échanger des informations et des idées (la base de l'informatique). A titre d'exemple, je travaille personnellement sur un générateur de source C à partir d'un fichier .BAT et sur la création d'une DLL Windows 3 permettant l'utilisation FACILE des DDE (avec gestion des modes d'échanges synchrones et asynchrones, des time-out etc...) et bien d'autres choses. Parallèlement, je suis en train de me mettre au courant du mode protégé des (3-4)86 Intel, car on peut en faire de bien belles choses. Même si les choses n'aboutissent pas entre nous comme je souhaite, je vous prierais de faire passer dans vos colonnes cette adresse : « Le monde en Tique, 18 rue Maître Albert, Paris 75005 ». Là se trouve sinon la meilleure, tout au moins une de mieux fournies des librairies informatiques de Paris (et je n'y ai aucune action!) ».***
Patrice Espié, Autrans

Donc note. Nous vous remercions de vos remarques et critiques, exprimés de la manière aussi franche - la spontanéité est une qualité rare. Quand au N° Hors Série, bien que nous comprenons que vous pouviez ne pas trouver, parmi les 23 programmes publiés, un qui vous convienne, il est difficile de commenter leur coût : cela reviendrait à 2,50 Frs le programme - allez trouver moins cher ailleurs! Par exemple, le programme « RETOUR AU DOS », qui permet de diviser la mémoire en deux parties, charger dans chacune un programme et basculer d'un à l'autre sur pression d'une touche. Un équivalent commercial s'appelle DoubleDos, coûte en USA 60$, en France (si on le trouve) quelque 950 Frs TTC. Ou celui qui efface sur disque tous les fichiers ayant une certaine extension, comme *.BAK (nous l'utilisons quotidiennement sur notre Réseau Local). En plus, tous sont livrés en source, donc modifiables et adaptables... Quand à la collaboration éventuelle, nous vous envoyons le dossier collaborateur. Un sujet comme DLL, DEE, OLE ou description des extendeurs du DOS existants, sont du plus haut intérêt. Nous attendons la proposition de vos textes.

☛ ***« Quelle ne fut pas ma surprise à la lecture du Hors Série n° 1! je m'attendais en effet, à y trouver les articles annoncés dans le n° 8. la plupart des articles annoncés ont effectivement été publiés, mais d'autres ne sont pas parus (suit liste). Certains paraissent intéressant et j'attends avec impatience de les voir au prochain sommaire. Je trouve par ailleurs votre revue fort intéressante du fait de son originalité par rapport aux autres revues d'informatique »***
Caroline Saussier, Versailles

Malheureusement, à cause de l'espace limité, même dans le Hors Série nous ne pouvions pas publier la totalité des textes qui nous ont été proposés. Nous avons choisi les articles les plus concis et du plus grand intérêt commun, pouvant être utiles au plus grand nombre de lecteurs. Le programme « Éditeur de palette écran » de M. Guere manque d'un texte explicatif (article); nous le publierons si l'auteur le complète. Les programmes « REVOIR », « PENDULE », « SOIR » de M. Mariaux manquent d'articles et de sources; nous ne pouvons pas publier sur ces pages des programmes binaires... Nous invitons l'auteur à nous envoyer les sources accompagnées des articles explicatifs. « Du problème posé à la solution programmée » de M. Riat s'est révéle faire partie d'un cycle d'initiation fort intéressant que nous avons choisi de publier dans les numéros normaux (précédent, actuel et prochain). « Accéder fichiers xBase en Pasca l» de M. G. Janot est toujours en réserve, le sujet a été traité dans l'article similaire de M. Janczak. « Compléteur de commandes » de M. El Guedj et « Interface conviviale » de M. G. Foucault sont arrivés sur des disquettes défectueuses, nous allons les publier dans le n° 15. « Capture de données » de Ph Dalléas convenait mieux au thème du numéro présent, consacré aux multi-tâches. En règle générale, les sujets ou choix annoncés, peuvent changer, suite à des impondérables comme l'actualité, contexte du numéro, espace disponible ou les préférences des lecteurs, signalés entre le temps...

ENTRE NOUS

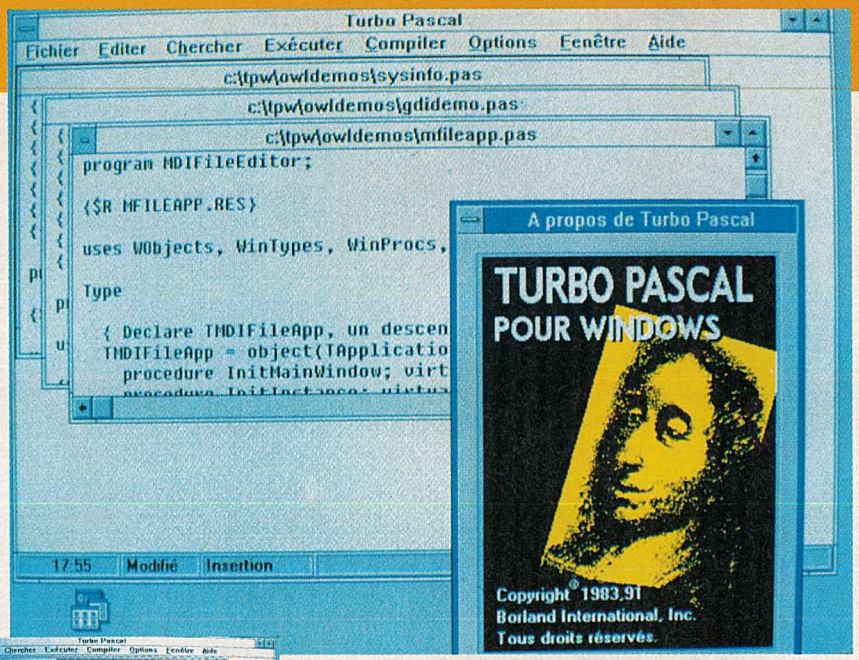
☛ ***« Je viens d'acheter votre revue, je pensais qu'elle était consacré aux ordinateurs AT286 et, déception malgré la qualité des articles, votre***

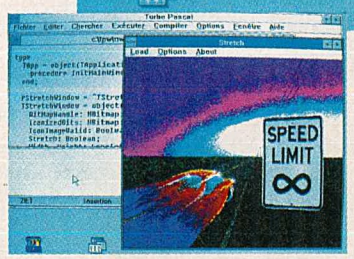
[SERVICE LECTEURS : INTERESSANT = 1 / MOYEN = 2/ MAUVAIS = 3]

TURBO PASCAL POUR WINDOWS

NE QUITTEZ PAS LE DOS SANS LUI

N'attendez plus pour obtenir le meilleur de Windows, rejoignez Borland, le leader en Programmation Orientée Objets et en programmation Windows. Avec Turbo Pascal pour Windows la création de vos applications Windows sera plus rapide et plus facile.
Turbo Pascal pour Windows inclut **GRATUITEMENT** la nouvelle structure d'application Object Windows. Maintenant, vous pouvez développer rapidement des applications Windows qui hériteront automatiquement du code des objets fenêtres, menus, dialogues, contrôles, et plus encore.





CRÉEZ DES APPLICATIONS WINDOWS SOUS WINDOWS.
L'environnement de développement intégré Windows vous permet de créer, éditer, compiler et exécuter vos programmes, le tout à partir de Windows.

CRÉEZ VISUELLEMENT DES RESSOURCES WINDOWS.
En utilisant la boîte à outils «Ressources» vous pouvez créer visuellement vos interfaces d'utilisateur Windows sans programmation.

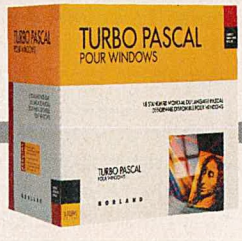
Créez des Applications Windows à moindre coût

Turbo Pascal pour Windows vous offre plus et vous coûte moins que les autres systèmes de développement Windows. Il est spécialement conçu pour la programmation Windows et tout ce qu'il vous faut est inclus pour un prix imbattable. Vous n'avez plus besoin d'acheter de Kit supplémentaire de développement. Turbo Pascal pour Windows est le moyen le plus facile de faire de votre prochain programme un programme Windows.

Adressez-vous à votre revendeur habituel ou appelez BORLAND au

N° VERT 05.46.96.69
APPEL GRATUIT

Utilisateurs de Turbo Pascal, appelez BORLAND pour votre mise à jour!



BEAC

BORLAND

Le Leader de la Programmation Orientée Objets pour Windows et DOS

43, avenue de l'Europe - BP 6 - 78143 Vélizy-Villacoublay - France - Tél.(33)(1) 39.46.96.69 - Télécopie (33)(1) 39.46.81.60 - Télex 698 793

revue s'intéresse de AT386. Je souhaite, si la revue était plus orienté PC286, m'abonner. J'espère que j'aurai une réponse me contenant ».
Jacques Tayssedre, Cergy

Bien que nous ayons consacré le dossier d'un numéro entier (7) à i386 et publié quelques articles consacrés spécifiquement à ce processeur, DEBUG n'est pas particulièrement dévoué à tel ou autre processeur. Il est vrai qu'au niveau Assembleur, nous nous concentrons de préférence sur la plate-forme Intel. Nous avons prévu aussi de publier un ou deux sujets intéressants sur les processeurs Motorola et RISC. Cela ne résout pas votre problème... La solution : tous les sujets réalisés en langages évolués (C, Pascal) et grande partie des sujets concernants les sujets « de base » Intel fonctionnent parfaitement sur votre processeur 286.

☛ ***« Suite à mon courrier, j'ai bien reçu un exemplaire de Dossier Collaborateur. J'ai écrit un utilitaire très puissant en Assembleur, programme dont le source comporte environ 25.000 lignes. J'aimerais savoir si je peux vous le soumettre ».***
Paul Rolland, Villeurbanne

Ce qui est plus intéressant dans le contexte d'une revue, est un bon article explicatif illustré par un court programme d'exemple qu'un long programme, consacré à une solution spécifique. Au fond, nous ne sommes pas des éditeurs de logiciels... D'autre part, pour des raisons techniques, il n'est pas possible de publier un listing qui dépasse 1.000 lignes. Déjà 500 lignes est une longueur considérable.

☛ ***« Je suis étudiant en informatique. Je vous écrit pour vous faire part de vos critiques envers DEBUG. D'abord, les côtes positifs (et il y en a beaucoup!). Je tiens tout d'abord à vous féliciter pour le redressement de situation dont vous avez fait preuve ces derniers mois. La revue est nettement plus intéressante et vos parutions plus régulières. Je me doute qu'avec une équipe aussi réduite, vous devez abattre un énorme travail. Je vous encourage donc à continuer car les lecteurs ne s'y trompent pas. bravo et longue vie à DEBUG! Voyons maintenant les côtes négatives. Il y a une chose que je ne comprends pas. Vous n'arrêtez pas de demander la participation de vos lecteurs, mais lorsqu'on essaye de vous contacter c'est le mur. En effet, je téléphone régulièrement pour vous proposer des idées, mais je n'ai aucune réponse. Je comprends fort bien que vous soyez occupé mais tout de même! Je voulais vous proposer mes services envers une rubrique biographique. Étant étudiant, j'ai accès à une très grande bibliothèque informatique ou sont présentes toutes les nouveautés. Je possède aussi une très grande quantité des procédures pures (en TURBO PASCAL), très pratique pour une foule de chose ».***
Bruno Pineda, Le Pontet

Ah, je voudrais pouvoir me couper en deux et pouvoir faire plus! Nous sommes trois : une personne qui s'occupe de l'administratif, une autre à la PAO et maquette, et votre serviteur pour le reste... Le mur existe effectivement, il est constitué des dossiers empilés, marqués « URGENT », « A FAIRE »,« PRIORITÉ », « IMPORTANT » etc. Enfin, comme je vous l' ai indiqué au téléphone, vos critiques biographiques ou revues de livres nous intéressent : 1 page par numéro, 2-3 livres, c'est à dire fichier ASCII ne dépassant pas 3000 caractères. Pour n° 15 la date limite est 25 juin. Et le sujet de la bibliothèque PASCAL à développer ou fractionner par sujet.

WANTED

☛ ***« Je me permets de vous envoyer ce courrier, afin de savoir s'il vous serait possible de trouver et publier un jeu de belote. Je pense qu'un tel programme intéresserait nombre de vos lecteurs, car il est actuellement impossible de trouver ce jeu dans le commerce ».***
Jean Pierre Eberlé, Schiltigheim

☛ ***« Je suis un très fidèle lecteur de votre magazine et ce depuis ses débuts. Je crois qu'il ne me manque que deux numéros. Je programme en C et en Basic, j'ai crée plusieurs programmes de gestion qui tournent sans problèmes chez des commerçants. J'ai acheté BASIC 7.1 de Microsoft et j'ai trop préjugé de mon anglais (toute documentation est dans cette langue). La manipulation des fichiers ISAM et grande partie de TOOLS me sont pratiquement inaccessibles par manque de traduction. Les exemples présentés sont complexes et on n'y trouve aucun organigramme. Au point de vue biographique, aucun livre ne traite de çe Basic 7.x en français (ni chez SYBEX, ni chez Micro Ap). quand à Microsoft .....!!!!! Alors, écrivez une série d'articles sur ces points (ou me donnez quelques conseils) ».***
François le Guelte, Marly le Roi

Nous n'avons rien de prêt pour répondre à ces deux sujets. Avis aux amateurs...

☛ ***« Il y a us sujet qui m'intéresse plus particulièrement et qui n'est que trop rarement traité dans les journaux spécialisés : l'intelligence artificielle, réseaux de neurones ».***
Joel Dubois, Fumel

Nous avons déjà deux ou trois articles de DDJ prêts. Le sujet mérite cependant le développement, contributions universitaires ou applications pratiques, réalisations utilisant les circuits spécialisés, comme ceux de chez Intel.

☛ ***« Un dossier sur la structure des fichiers TIFF, GIF ... etc ».***
Michel Gonzalez, Merignac

☛ ***« TIFF vous connaissez ? Je dispose d'un certain nombre d'images provenant d'un scanner que j'aimerais pouvoir relire par un programme. Mais je ne connais pas le format d'enregistrement. Je le recherche désespérément depuis des semaines. Pourriez-vous m'aider ? »***
Fabrice Casoni, St Genevieve des Bois

Contribution américaine introuvable. Recherchons les auteurs compétents.

☛ ***« Avant tout apparier l'orientation que vous avez choisi à savoir être un magazine pour les programmeurs. Je dois même vous féliciter car vous parvenez à intéresser les plus chevronnés (sans vouloir me vanter!). Vos articles sont vraiment clairs et concis et d'un style fort plaisant à lire. Vous n'avez jamais étudié le problème de la compression de fichiers, des algorithmes ...»***
Hervé Ronin, Sausset les Pins

Nous avons trouvé un choix d'articles US. Cherchons contributions françaises. Sujet prévu pour septembre (?), en tout cas à la rentrée.

DEBUG MAGAZINE

Rédacteur en Chef
Directeur de Publication
Artur G. Piotrowski

Maquettiste
Elisa Le Moal

Ont collaboré à ce numéro
ART, Serge Grun,
Jean Christophe Riat, Tom Green,
Allen Hollub,
Margaret Johnson, Mark Solinski,
G. Korwin, Jean Pascal Cote,
Philippe Dalléas.

Iconographie
Photos : Borland, Microsoft,
Pictor International, Debug Magazine,
Wordstar.

REDACTION
TELEMATIQUE
VENTE AU NUMERO
SERVICE LECTURE
GESTION DES ABONNEMENTS
EDITEUR
SOFTMEDIA
SARL de presse au c^al de 5000 F.
14, rue de l'Atlas
75019 Paris
Téléphone : (1) 44 52 03 33
Télécopie : (1) 44 52 03 29

Coordination Production
Christian Hernault

COMPTABILITE - ABONNEMENTS
Lidia Zakar.

Prix de l'abonnement
11 numéros + 1 numéro spécial
486 F. T.T.C. (France)
Prix au numéro : 37 F.

IMPRESSION - BROCHAGE
La Presse de Bretagne

Mise en page : X.PRESS-Macintosh

Dépôt légal à la parution
Commission paritaire 71290
ISSN 0998-9250
membre inscrit à O.J.D.

« La loi du 11 Mars 1957 n'autorisant pas aux termes des alinéas 2 et 3 de l'article 41 d'une part que les copies ou reproductions strictement réservées à l'usage privée du copiste et non destinés à une utilisation collective et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, faites sans le consentement de son auteur ou de ses ayant droit ou ayant cause, est illicite (alinéa premier de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du code pénal.

SOMMAIRE

N° 14 - Juin 1991

EDITO 1

PARLONS 2

RUMEURS 8

DOSSIER Dr. DOBB'S
Un noyau multi-tâche en C++ 13
Un noyau multi-tâche fondé sur le principe de « préséance » 16
Les bibliothèques de liaisons dynamiques DLL sous Windows 24

PROGRAMMATION
Environnement Multi : découverte de l'Assembleur 29
Cahier Listing 31
Windows 63

INITIATION
La récursivité démystifiée 51

DESSIN
Cup of Tea 55

EVALUATION
Accélérez votre PC 58

NOUVEAUX PRODUITS : Le Dos nouveau est arrivé 70
PODIUM : Nouvelle énergie pour les stars 72
MESSAGES 77
ANNONCES 78

LA PAGE DE JOHN C. DVORAK 80

☛ ***« J'ai fait le test MFlop de Microsystemes sur mon SX avec TC 2.0 et C++, tout en prenant le soin de garder les mêmes réglages. Conclusion : j'obtient une baisse de 26% de performances entre TC 2.0 et TC ++. Plus encore, en remplaçant DOS 4.01 par DR DOS 5.0, j'obtiens par ce même test une baisse de 57% des performances ! D'autre part, j'ai apprécié vos articles sur CHAOS et fractales et je serais intéressé par une version Assembleur du tracé d'ensemble de Mandelbrot ».***
Christophe Laskawiec, Castres

Votre remarque sur DR DOS se confirme, sans que personne n'arrive pas à trouver la raison (voir aussi la note en bas de l'article sur accélérateurs de disques dans ce numéro). Le dossier sur Chaos a beaucoup plu à tout le monde. Allons essayer de revenir à ce sujet, entre temps vous pouvez trouver beaucoup de nouveaux livres consacrés à Chaos en librairies.

BUGS DANS DEBUG ?

« Dernièrement, chez mon libraire, je remarque votre n° 12, avec page 26, sujet : Protect et Vérif. Ayant moi-même mis au point un programme de cette sorte, j'achète. Je tape le programme page 44 : PROTECT. Résultat : un magnifique plantage, mais bien planté, obligé de rebooter. Le programme ne marche pas. Alors que faire? 37 Frs, listing inutilisable, disquette vendue ? J'ai payé pour rien. J'espère que vous pourrez faire quelque chose ».
Raphael Berg, St Max

Nous sommes littéralement stupéfaits par l'intérêt que ce sujet a éveillé! Quand à votre problème, il peut venir du format de disquette utilisé, configuration du système ou paramètres du DOS. Vous ne les indiquez pas! La version de M. Vasseur a été compilée et vérifiée plusieurs fois - elle marche. Nous vous conseillons de contacter directement M. Vasseur, pour précisions et détails.

☛ ***« Ayant acheté le n° 9, j'ai eu la désagréable surprise de constater que le listing du programme d'émulateur Minitel en C est incomplet, il manque la procédure Réception ».***
Sylvain Carretero, Chevigny Saint Sauver

Le programme a été développé pour et compilé avec Turbo C 2.0 et Turbo Assembleur 1.0 Borland - les procédures manquantes se trouvent dans la bibliothèque standard. Nous suspectons que vous utilisez le C de Microsoft. Les procédures manquantes n'existent pas dans la bibliothèque fournie avec les langages Microsoft - une adaptation est nécessaire.

COMMENT PRESERVER SON IMAGE

☛ ***« Abonné à DEBUG MAGAZINE et appréciant la diversité ainsi que la qualité des contenus, je me permets de faire appel à la compétence d'un rédacteur concernant un problème que je n'arrive pas à résoudre. En effet, je souhaite sauvegarder sur disque une image graphique couleur en 640x480; le graphique étant crée en Turbo Pascal 5.5 au moyen de procédures de GRAPH.TPW. Mais je souhaite surtout récupérer le fichier image ainsi crée avec les couleurs d'origine ».***
Jean Pascal Molina, SEYSSINS

D'après votre description, je suppose que vous utilisez votre carte dans le mode vidéo 12h (640x480, 16 couleurs). Dans ce mode, la mémoire vidéo commence effectivement à l'adresse A000:0000. Cette mémoire est organisée en quatre pages, qui ont toutes les quatre la même adresse physique. La combinaison des quatre mêmes bits provenant de quatre octets situés à des adresses identiques dans les quatre pages forme un demi-octet qui sert de pointeur dans la palette par défaut. Dans le mode par défaut, c'est à dire en utilisant la palette telle qu'elle a été initialisée par le BIOS, ces quatre plans correspondent aux trois couleurs primaires (RVB) plus l'intensité (ce n'est plus exact dans les modes étendus comme, par exemple, 800x600 en 256 couleurs). Vos problèmes proviennent du fait que votre routine ne sauvegarde qu'une seule de ces pages et qu'à la lecture, les données lues sont chargées simultanément dans toutes les pages. Vous avez supposé que les huit bits d'un octet correspondent à huit pixels consécutifs et que 80 octets consécutifs forment une ligne de 640 pixels, ce qui est effectivement exact. Toutefois, la déclaration de la page de mémoire vidéo sous la forme:

```
T_Ecran:Array[0..16383] of Word
```

est inadéquate: elle correspond à 32768 octets, alors qu'une page de 640x480 pixels a une taille de 38400 octets (en comptant huit pixels par octet). Par ailleurs, votre routine ne sauvegarde qu'un seul plan. Pour sauvegarder les quatre plans, il eût fallu placer l'instruction d'écriture à l'intérieur de la boucle « For I:=0 to.. ». En outre, je ne saisis pas très bien l'utilité des boucles « For Col...:=» et « For Ligne:=...», ainsi que de la ligne:

```
Octet:=Octet or Mem[$A000:Ligne*80+Col];
```

Je vous propose donc la solution suivante:

```
Assign(Cliche,Nom_Fichier);
Rewrite(Cliche);
P_Cliche:=Ptr($A000,$0); {début de la page vidéo}
For I:=$0 to $3 do {pour chacun des 4 plans}
  Begin
    PortW[$03CE]:=$0004+I*256; {activation du plan I}
    Write(Cliche,P_Cliche^) {et sauvegarde}
  End;
Close(Cliche);
```

Le port 3CEh permet de choisir le registre qui sera lu ou écrit à travers le port 3CFh. L'écriture de 04h (autrement dit l'activation de « Read Map Selec t») à l'adresse 3CEh ouvre ainsi accès, par 3CFh, à un registre qui contrôle quelle(s) page(s) de la mémoire vidéo sera disponible en lecture à l'adresse A000:0000. Il suffit donc d'écrire successivement les valeurs 00h à 03h à l'adresse 3CFh pour pouvoir accéder aux quatre pages. L'inconvénient de cette technique est que chaque écran sauvegardé fait exactement 150 kilo-octets... Je ne saurais trop vous encourager à utiliser une quelconque méthode de compression (RLE - Run-Length Encoded, LZW - Lempel-Ziv-Welch ou encore Huffmann). Le RLE devrait être relativement rentable, même si vous ne compressez l'image qu'octet par octet (et non pas bit par bit). Ces deux méthodes ont été décrites, l'une dans un numéro récent de Pascalissime, l'autre dans le numéro 156 du « Dr Dobb's Journal ». La restauration de l'image est à peine plus compliquée. Là aussi, il faut prendre soin de désigner la page mémoire dans qui doit recevoir les données lues. Le port $03C4 permet d'accéder, à travers l'adresse $03C5, au registre de sélection du plan mémoire adressé en écriture. Les bits 0 à 3 désignent les plans mémoire respectifs - chaque bit égal à 1 autorise l'accès au plan correspondant. Je vous propose donc la routine suivante:

...Suite p. 76

CODE C POUR PC

code source inclus, bien entendu

	PRODUIT	DESCRIPTION	PRIX EN $ US
		MS-DOS FILE COMPATIBILITY PACKAGE (crée, lit et écrit les fichiers MSDOS sur les ordinateurs non-MSDOS)	750
		M++ (mathématiques rapides, classes C++, indexation, matrices, vecteurs, numériques, gestion mémoire, E/S; spécifier le compilateur C++: Turbo/ZORTECH/GLOCK)	395
		C++/VIEWS (interface C++ à WINDOWS 3.0; plus de 60 classes, browser gratuit (sans source), pas des royalties. Seulement pour ZORTECH C++)	395
MISE A JOUR !		GRAPHIC 6.0 (haute résolution, graphiques scientifiques en couleurs avec recopie papier, dessin de contours, indépendant de matériel)	370
		MBALLOC (le meilleur gestionnaire du heap, remplace les routines standard, donne les performances Microsoft avec Turbo C)	340
		VICTOR IMAGE PROCESSING LIBRARY & ZIP IMAGE PROCESSOR (luminosité, contraste, fusion d'images, binaires TIFF/GIF/PCS, compatible Scan Jet)	290
		TURBO TEX (version 3.0; HP, PS, imprimantes matricielles, fontes CM, La TEX, MetaFont)	250
		DB-FILE & DB-RETRIEVE BUNDLE BY RAIMA (base de données B-tree et Réseau avec query et reporting SQL (version multi-utilisateurs $ 475)	245
		TE EDITOR DEVELOPER'S KIT FOR WINDOWS (éditeur plein écran, commande UNDO, fenêtres multiples, inclus TER pour applications incorporées, pas de royalties)	220
		CODE BASE 4 (gestionnaire base de données, indexes compatibles dBase et Clipper, version 4.2)	210
NOUVEAU!		STRING EXTERNALISATION TOOLS (un .EXE, plusieurs langues. Changee la langue de prompteurs et messages en remplacant le fichier exterieur. Pas de royalties.)	210
		INSTALL 3.05 (programme d'installation automatique, installation partielle au choix d'utilisateur, vérification CRC)	195
NOUVEAU!		C-PSLIB (Bibliothèque de génération du PostScript pour les programmes en C; inclut graphiques completes, fontes, rotations et support de paragraphes)	170
		REPORT RIGHT (générateur de rapports et graphiques pour WINDOWS 3.0)	170
		C-INDEX/PC DATABASE LIBRARY (indexation B+tree rapide, chaînes/binaires/clés sur mesure, formats multiples d'enregistrements, manuel 170 pages)	165
		ROGUE WAVE TOOLS.H++ OR MATH.H++ CLASS LIBRARY (documentation extensive)	CHAQUE 165
		DXF SUBROUTINE LIBRARIES (outils pour créer les fichiers d'images AutoCAD DXF; manuel et programmes simples inclus)	155
		WKS LIBRARY VERSION 2.01 (interface programme C pour Lotus 1-2-3, dBase, Supercalc 4, Quattro et Clipper)	155
		RED (éditeur de développement pour WINDOWS, enregistrement de frappe clavier avec rappel, UNDO illimité)	150
		MINIX OPERATING SYSTEM (version 1.5, système opérationnel style UNIX, manuel inclus. Précisez disquettes 5" ou 3")	CHAQUE 150
NOUVEAU!		DELORIE GCC FOR MS-DOS (version 1.39. Inclut : C++, extendeur pour l'Assembleur et DOS, source complet et fichier make)	150
		VMEM/C (gestionnaire de la mémoire virtuelle. Dernière page utilisée ou pages récentes. Expansion dynamique du fichier de swap.)	140
		CBTREE (Pilote B+tree ISAM. Clés multiples de longueur variable.)	135
		TE EDITOR DEVELOPER'S KIT (Editeur pleine page. Commande UNDO. Fenêtres multiples.)	130
		C COMMUNICATIONS TOOLKIT BY MAGNA CARTA (Adapté aux cartes multi-port, avec coprocesseur, cartes télécopie. Géré par interruptions. Emulation protocoles xfer.)	125
		C++ OBJECT LIBRARY (Fenêtres virtuelles, E/S, listes, gestion d'espace-fichiers libre, E/S fichiers ISAM par clé.)	125
		SILVER WARE "C" EMM LIBRARY (60 fonctions d'interface à EMM. Version 4.0)	120
		WINMEM (Handles globals illimités pour WINDOWS 3.0. Bloc d'overheaper de 4 octets au lieu de 20. Mémoire virtuelle pour 286.)	110
		NETWORK TOOLBOX/N (Les fonctions C pour Novell NetWare 2.1x. Programmes simples inclus.)	110
NOUVEAU!		SCM (Schéma portable en C, conforme aux spécifications IEEE et 3.99. Ramassage de résidus de la pile C pour mélange facile avec C.)	100
		PC/IP (CMU/MIT TCP/IP pour pilotes PC, Ethernet, Appletalk, NetBios; pilotes Clarkson, clone de NFS de Stan, expéditeur Bdale.)	100
		POWERSTOR (version 1.2. Espace étendue sur le heap dans la mémoire expansée, étendue et/ou sur le disque dur.)	95
		MIRACL MULTIPRECISION INTEER & RATIONAL ARITHMETIC C LIBRARY (Près de 100 fonctions. C&C++ en provenance d'Irlande.)	90
		DISASSEMBLER (8086/80286/80386/80C186 avec 80287/80387/80C187 NPX)	90
		BOOTER TOOLKIT (Routines de démarrage pour disquette souple, système de fichiers DOS, multi-tâche léger, gestion de mémoire rapide.)	85
		VM4C (Gestionnaire de la mémoire virtuelle pour C. Commute blocs de mémoire sur disque. Petit encombrement, rapide, simple.)	80
MISE A JOUR !		VMEM (Gestionnaire visuel de mémoire par Blake McBride, version 3.6. Pagineur LRU, fichier de commutation dynamique, sauvegarde/restauration d'image)	80
		MULTIDOS PLUS (Multi-tâche basé sur DOS. Messagerie inter-taches, sémaphores.)	80
		BGI PRINTER DRIVER TOOLKIT (Pilotes BGI pour Epson 9 et 24 aiguilles, HP Laser Jet II et Paint Jet.)	75
		COMM-DRV (Communication série gerée par interruptions, pilotes, moniteurs connectés aux ports série. Source uniquement avec bibliothèques; source complet $ 200)	75
		FLEXLIST (Multiples méthodes d'accès aux listes de données arbitraires, doublement linkées. Spécifier C ou C++.)	CHAQUE 65
		FOUNDATIONS 1 C++ CLASS LIBRARY (Enregistrements E/S en ASCII, matrices binaires, gestion d'exceptions, B-tree, objets persistants.)	60
NOUVEAU!		C-LIN (Bibliothèque de l'algèbre linéaire. 42 fonctions matricielles et vectorielles, inversion et décomposition de matrices, solution d'équations.)	60
		MULTISTR (Evaluation de chaînes contenant multiples expressions arithmétiques ASCII avec variables.)	60
		CODER'S PROLOG (Version 3.0. Moteur d'inférence pour utilisation avec programmes en C.)	60
		FINAN C (Grande collection des fonctions financières, incluant obligations, inventaire, portefeuille d'actions et cash flow.)	55
		PASCAL P-CODE COMPILER & INTERPRETER OR PASCAL-TO-C TRANSLATOR (Niveau 0 ISO standard avec quelques caractéristiques du niveau 1.)	CHAQUE 50
		CALC (Evaluation d'expressions algébriques ASCII, imbrication de parenthèses illimité, symboles, 32 fonctions incorporées, facilement extensible.)	50
		BACKUP & RESTORE UTILITY BY BLAKE MCBRIDE (Volumes multiples, compression de fichiers, encryptage.)	50
		FLOPPY TAR (Sauvegarde et restauration TAR sur les unités MS DOS. Accès direct aux unités non-standard.)	50
NOUVEAU!		C COMPILER PACK (5 compilateurs C : 3 pour 8086 (gcc, MicroC, Small C), 2 pour 68000 (Sozobon, cc68), portage gcc inclut seulement source de la bibliothèque.)	50
NOUVEAU!		GNUISH MS-DOS (Portage de 17 programmes GNU à MSDOS organisé par Thorsten Ohl.)	50
		SUPERGREP (Algorithme de recherche exceptionnellement rapide, recherche de texte révolutionnaire, recherche aussi les sous-répértoires.)	50
		OBJASM (Conversion des fichiers .obj en .asm. Sortie compatible MASM.)	50
		YATE (Encore Un Editeur de Texte ("Yet Another Text Editor"). Ecrit en C++, plein écran.)	50
		NIH CLASS LIBRARY & BOOK (Classes de base C++ et le livre "Data Abstraction and Object Oriented Programming in C++" par Keith Gorlen.)	50
		KIER DATELIB (Tout type de manipulation de la date : translation, validation, formatage, calculs arithmétiques.)	45
		RXC & EGREP VERSION 2.0 (Compilateur des Expressions Régulières et comparaison de formes, machine d'état fini à partir des expressions régulières.)	35
		BISON & BYACC (Générateurs d'analyse de type YACC, documentation, pas de restrictions dans l'usage du produit du BYACC.)	35
		PC-XINU (Système XINU pour PC, par Comer.)	35
		SPELL PACK (6 vérificateurs d'orthographe, césure, 2 ensembles d'utilitaires, vocabulaire 60Ko, Ispell, Microsp, Sp, Cspella, Spell, Dawg, Soundex)	30
		REGX PLUS (Routines de recherche et remplacement de chaînes, basées sur les expressions typiques.)	30
		GNU AWK & DIFF FOR PC (Deux programmes reunis dans un seul ensemble.)	30
MISE A JOUR !		6-PACK OF EDITORS (10 éditeurs du domaine public, pour l'usage, étude ou hacking. Sont inclus : microEmacs 3.10, Stevie, Elvis et Origami.)	30
		CRUNCH PACK (25 programmes de compression et expansion de fichiers.)	30
		MAKE PACK (8 versions de Make incluant GNU Make, Cake, Gymake et makefile maker.)	30
NOUVEAU!		BIGNUM (Ensemble d'arithmétique avec arbitrage de la précision, portable et efficace. Manuel imprimé. Provenance française.)	25
		STRING PACK (Multitude de routines de manipulation de chaînes. Classe chaînes C++, BAWK, césure mots, recherche approximative, Booyer-Moore, Hypertext etc.)	25
		PC-MAIL (Ensemble UUCP par Wietse Z. Venema. Réception, envoi et gestion du courrier UUCP. Provenance : Pays Bas.)	25
		PERL FOR MS-DOS (C, sed, awk et shell, tous incorporés en un langage. Manuel imprimé inclus.)	25
		FLEX (Générateur rapide d'analyse lexicale. Nouveau LEX amélioré. Version 2.3.6 avec doc.)	25
		USING C++ LIBRARY (Le code correspondant au livre, par Bruce Eckel. Compatible ZORTECH 2.0)	25
		GNU CHESS (Version 3.1 GNU et version 1.01 Windows. Exécutable inclus. Nécessite SDK pour assembler. Vraiment beau!)	20
		GNU RCS (version FSF du "Revision Control System". Comme SCCS d'UNIX mais meilleur, maintien la trace de développement des logiciels.)	20
		SDBM (Gestionnaire rapide de la table hachée basée sur disque, pour tables réellement larges. Clone de ndbm d'UNIX.)	20
		VAL LINKER BY DAVID TROENDLE (Linkeur complète. Excellent pour étudier le linkage.)	20
		NLMDL (Remplissage des carrées non linéaires en C++.)	20
NOUVEAU!		X-WINDOWS (Client et Serveurs, bibliothèque partagée, Xlib, exécutables, XAW etc. Diffs en X11R4; spécifier TCP ou la même machine.)	200
NOUVEAU!		KYOTO COMMON LISP (Parfum d'Austin, naturellement. Version 1.530; inclus GCC et Portable Common Loops (PCL).)	140
		GNU EMACS VERSION 18.55 (Source complète et exécutables inclus.)	125
		WAFFLE BBS (Interface à UUCP et B ou C News. Forums internes. Section fichiers. Traitements externes. Support complet d'USENET.)	120
		GNU C COMPILER (GCC) VERSION 1.39 (Source complet et exécutables inclus.)	100
		VIEWCOMP SPREADSHEET (Termcap interne, sortie imprimante, manuel imprimé.)	80
		GNU DEBUGGER (GDB) VERSION 3.5 (Source complet et exécutables inclus.)	60
		GILLESPIE PASCAL-TO-C (Plusieurs variantes de Pascal traités. Sortie lisible et maintenable.)	25
		GOSLING SPREADSHEET PACK (Variante de tableur Gosling de 1982, incluant PubliCalc et SC version 6.10. Variante MS DOS.)	25

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3587
USA

Téléphone : 19 - 1 (512) 258 0785
Télécopie : 19 - 1 (512) 258 1342
E-mail : info@acw.com

Tous les produits en anglais. Paiement par EuroCard/Visa (indiquer le numéro et la date d'expiration), Mandats Internationaux ou chèques sur une banque américaine.

SERVICE LECTEURS : 2

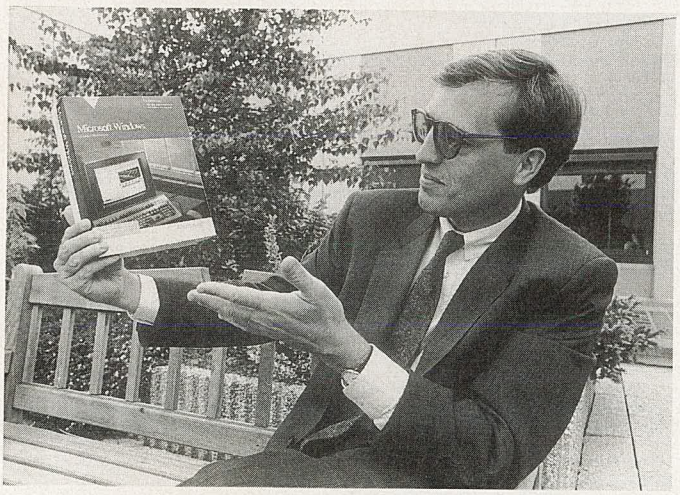
RUMEURS

TENDANCES ET ASPECTS DE L'INDUSTRIE

par *ART*

⇨ LA CRISE ? QUELLE CRISE ?

Notre industrie est en train de vivre une révolution, peut être la plus importante depuis dix ans. Cette fois ci, ce sont les géants qui s'affrontent, des compagnies qui « pèsent » des centaines de millions ou des milliards de dollars - ce qui ne préjuge pas du vainqueur. La lutte se déroule sur plusieurs champs à la fois :



**« Le grand vainqueur :
WINDOWS présenté par le directeur de Microsoft France,
Michel Lacombe »**

Le premier champ est celui de systèmes opérationnels. Les trois acteurs principaux sont : IBM (OS/2), Microsoft (MS DOS et Windows) et Digital Research (DR DOS et Multiuser DOS). L'enjeu est le remplacement de quelque 60 millions (dont 3.5 millions en France) des DOS poussifs par un système adapté à de nouvelles machines et plus convivial (c'est à dire multi-tâche et graphique). Depuis un an, Digital Research n'a pas réussi à entamer sérieusement la forteresse Microsoft, ne vendant que quelque 60.000 exemplaires du DR DOS dans le monde entier. Ce n'est qu'une goutte dans l'océan, la grande vague attend sans doute MS DOS 5, qui a été officiellement présenté par Bill Gates le 11 juin passé devant Boston Computer Society. Mais DR n'entend pas se laisser faire dépasser - la présentation officielle de MS DOS 5 va relancer le marché, actuellement bloqué par l'attentisme. Il va contre-attaquer, sans doute à la rentrée, avec une nouvelle version du DR DOS, encore plus performante, apportant notamment un meilleur cache, un compacteur automatique de fichiers et une meilleure gestion de tâches. Dans le domaine des systèmes multi-tâche, on ne voit pour l'instant le vainqueur à coup sur: OS/2 d'IBM sera concurrencé par la version 32 bits Nouvelle Technologie de Microsoft, attendue quelque part entre 1992 et 1993. Les deux versions restent trop lourdes, trop compliquées, trop chères et incompatibles avec les applications DOS pour devenir populaires. Microsoft a vendu entre 3 et 4 millions d'exemplaires de Windows pendant qu'IBM n'a vendu que 300 mille d'OS/2 au total. L'outsider, Digital Research, manque de poids pour s'imposer à l'industrie, bien que son Multiuser DOS marche bien, est compatible, remarquablement rapide et simple à utiliser. La réconciliation va venir peut être d'un système unique, multi-tâche, performant et léger, pourvu d'une interface compati-

.....TELEX.....

➦ **BONNES AFFAIRES.** Si vous avez NetWare de Novell, contactez d'urgence un des revendeurs, tels qu' INTERQUADRAM à Boulogne (1) 46840515. Novell mène actuellement une importante campagne de mise à jour qui vous permettra d'actualiser votre système à bas prix.

➦ **MODEM RAPIDE.** Jusqu'au 30 juin Olitec propose sa nouvelle carte modem V32/V32bis MNP5/V42bis à 7 100 Frs (5 990 Frs HT). C'est (presque) le prix américain, pour le produit pourtant français. Profitez de cette offre en appelant 83 21 95 15.

➦ **NOS IDEES ONT DES SUITES. ONTRACK,** spécialiste de logiciels utilitaires pour disques durs et pour NetWare de Novell annonce pour septembre NETOptimizer, un logiciel d'optimisation pour les volumes sous NetWare (voir rubrique IDÉES du DEBUG MAGAZINE n° 8). Dommage, que ce soit outre-atlantique. Ontrack au 1 (612) 9371107

➦ **IBM** prévoit lancer un portable basé sur la technologie PenPoint de GO Corp. à la fin de l'année.

➦ Si vous voulez muscler votre 286, contactez Kingston Technology à Fountain Valley, en Californie. Cette entreprise présente une plaquette 386SX qui se branche directement dans le support de 286. La version 16 MHz coûte 645$, la version 20 MHz coûte 695$. SX/Now, car tel est le nom de la plaquette, possède un support pour le coprocesseur 387SX. Téléphone : 19 1 (714) 435 2600

➦ **Cache Computers de Fremont** en Californie présente « la plus rapide carte mère i486 ». Le processeur, coiffé d'un Icecap CPU de Velox Corp, fonctionne à 50 MHz. La carte BAT 486-50/AT coûte 3950 $ US. téléphone : 19 1 (415) 226 9922. Les réalisations similaires ont été présentées aussi par IBM, Arche et Dell.

➦ **Phar Lap** annonce une nouvelle version de 386/DOS Extender, compatible WINDOWS et DOS 5.0. Le prix est de 495$ US.

[SERVICE LECTEURS : INTERESSANT = 4 / MOYEN = 5 / MAUVAIS = 6]

pliez ici en second et collez l'autre côté au ruban adhésif.

AFFRANCHIR
AU TARIF
LETTRE..

DEBUG MAGAZINE
14, rue de l'Atlas
75019 Paris

pliez ici en premier.

QUESTIONNAIRE – LECTEUR

Année de naissance : 19__
Sexe : ☐ M ☐ F

Votre secteur d'activité :

- ☐ Agriculture
- ☐ Enseignement
- ☐ Autres Fonction Publique, Collectivité locale
- ☐ Association
- ☐ Energie
- ☐ Métallurgie
- ☐ Chimie, verre, plastique
- ☐ Construction automobile, mécanique, navale, aéronautique
- ☐ Electronique
- ☐ Bâtiment, travaux publics
- ☐ Autres secteurs industriels
- ☐ Commerce
- ☐ Transports
- ☐ Télécommunications
- ☐ SSII
- ☐ Autres services aux entreprises
- ☐ Hôtellerie, réparation et autres services aux particuliers
- ☐ Banque, Assurance
- ☐ Autres secteurs commerciaux

Nombre de salariés dans l'établissement : ____

Votre profession :

- ☐ Propriétaire
- ☐ PDG-DG-Gérant
- ☐ Cadre
- ☐ Employé
- ☐ Ouvrier
- ☐ Retraité
- ☐ Autre

Votre matériel :

- ☐ IBM ou compatible
- ☐ Apple, MAC
- ☐ Mini
- ☐ Mainframe

L'utilisation de votre matériel :

- ☐ privée
- ☐ professionnelle
- ☐ les deux

Etes-vous satisfait de votre équipement actuel ? ☐ Oui ☐ Non
Allez-vous l'étendre ? ☐ Oui ☐ Non

Vos commentaires :

BULLETIN D'ABONNEMENT *2 ANS COMPLET*

❒ Je m'abonne pour **22** numéros + **24** disquettes listing + **2** numéros Hors - Série : **982 F** au lieu de **1 964 F**

Je règle par : ❒ chèque à l'ordre de **Debug Magazine**

❒ carte bancaire

- 50 %

Nom de la carte : ....

N° .../.../.../.../.../.../.../.../.../.../.../.../.../.../.../.../ Date d'expiration /.../.../.../.../.../.../.../.../.../.../.../.../.../.../.../.../

Nom : .... Prénom : ....

Société : ....

Rue : ....

Lieu : ....

Code postal : .... Ville : ....

Pays : .... Tél : ....

Date : .... Signature : ....

L'abonnement commencera au n° ....

A renvoyer à **DEBUG MAGAZINE** - 14, rue de l'Atlas - 75019 Paris

---

BULLETIN D'ABONNEMENT *2 ANS SIMPLE*

❒ Je m'abonne pour **22** numéros + **2** numéros spéciaux : **512 F** au lieu de **930 F**

Je règle par : ❒ chèque à l'ordre de **Debug Magazine**

❒ carte bancaire

- 45 %

Nom de la carte : ....

N° .../.../.../.../.../.../.../.../.../.../.../.../.../.../.../.../ Date d'expiration /.../.../.../.../.../.../.../.../.../.../.../.../.../.../.../.../

Nom : .... Prénom : ....

Société : ....

Rue : ....

Lieu : ....

Code postal : .... Ville : ....

Pays : .... Tél : ....

Date : .... Signature : ....

L'abonnement commencera au n° ....

A renvoyer à **DEBUG MAGAZINE** - 14, rue de l'Atlas - 75019 Paris

---

BULLETIN D'ABONNEMENT *1 AN SIMPLE*

❒ Je m'abonne pour **11** numéros + **1** numéro spécial : **279 F** au lieu de **465 F**

Je règle par : ❒ chèque à l'ordre de **Debug Magazine**

❒ carte bancaire

- 40 %

Nom de la carte : ....

N° .../.../.../.../.../.../.../.../.../.../.../.../.../.../.../.../ Date d'expiration /.../.../.../.../.../.../.../.../.../.../.../.../.../.../.../.../

Nom : .... Prénom : ....

Société : ....

Rue : ....

Lieu : ....

Code postal : .... Ville : ....

Pays : .... Tél : ....

Date : .... Signature : ....

L'abonnement commencera au n° ....

Anciens numéros à **35 F** pièce : ❒ n° 2 ❒ n° 3 ❒ n° 4 ❒ n° 5 ❒ n° 6 ❒ n° 7 ❒ n° 8 ❒ n° 9

Anciens numéros à **37 F** pièce : ❒ n° 10 ❒ n° 11 ❒ n° 12

Hors - Série n° 1 à **58 F**

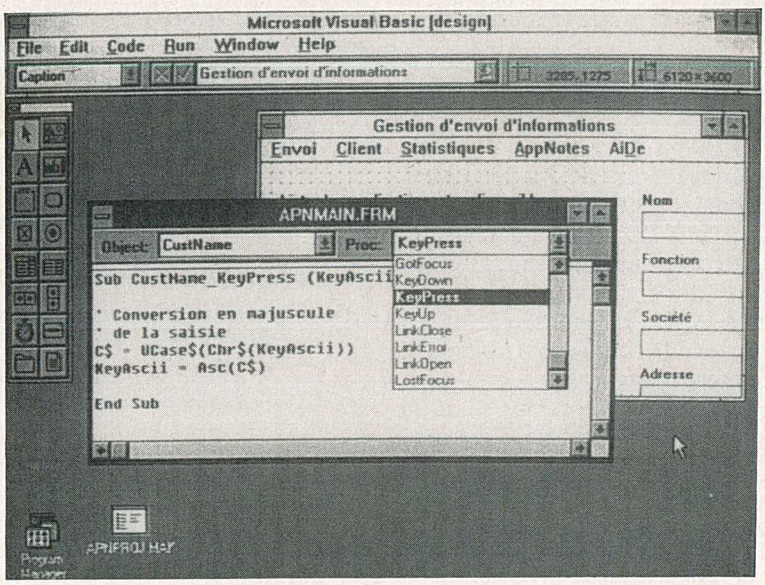
A renvoyer à **DEBUG MAGAZINE** - 14, rue de l'Atlas - 75019 Paris

tible WINDOWS - le seul standard autour duquel l'industrie commence à graviter d'un commun accord. Le nom proposé est WIN DOS (ou VIN DOS, DOS Vainqueur). A qui les honneurs de la première?

L'enjeu va cependant plus loin. Apple vient justement de commencer à livrer le Système 7, longtemps attendu. Les grandes manoeuvres de mises à jour et de compatibilité des anciennes applications commencent. Pour promouvoir le nouveau système Apple prévoit de dépenser, pour les seuls USA, 20 millions de dollars sous le slogan « la prochaine machine DOS que vous devez acheter » (« Next DOS machine you should buy ») qui vise directement ses principaux concurrents. L'objectif est d'avoir 3 millions d'utilisateurs sous Système 7 dans un an. Mais de plus en plus d'acheteurs se posent la question : pour le nouveau matériel, vaut-il mieux investir dans Mac, qui reste toujours horriblement cher, ou dans une solution com-patible IBM, ou on trouve de plus en plus d'applications graphiques, d'une facilité d'utilisation comparable à Apple. La firme de Cupertino comprend bien cette menace. Justement, devant le tribunal qui doit se pencher sur sa plainte contre Microsoft et Hewlett Packard, ses avocats ont fait état des « menaces et mensonges » qu'aurait utilisé Microsoft pour obtenir la licence de technologie Windows en 1985. Apple a aussi élargi sa plainte sur WINDOWS 3. La contre-attaque a lieu aussi sur le champ technologique : Apple annonce l'investissement pour promouvoir le développement des applications S7 en Europe.

Le deuxième champ de bataille, est celui des tableurs. Trois poids lourds : Lotus, Microsoft et Borland s'affrontent pour récupérer le marché estimé à quelque 10 millions d'utilisateurs. En attendant, pendant que les uns (Microsoft) livrent tranquillement ses superbes tableurs (Excell, le seul tableur livrable actuellement et exploitant Windows à fond), les autres (Lotus) nous montrent des maquettes (1-2-3 Win, actuellement en alpha), dans lesquelles nous allons trouver (quand ce sera livrable) les fonctionnalités déjà installées chez les un (Excell) et dont on s'emploie furieusement à terminer l'implémentation. Le problème de Lotus est que 1-2-3 Win n'est guère avancé : on nous le promet pour septembre. En attendant, les campagnes de remplacement (« mise à jour ») de Borland semblent ébrécher sérieusement le gâteau, jusqu'à là quasiment monopolisé par Lotus. Borland annonce avoir vendu quelque 1 million d'exemplaires de Quattro, elle avance à grands pas (actuellement la promotion de la version 3), tout en promettant la version WINDOWS, aussi pour la rentrée.

Le troisième champ, celui des bases de données, semble pour l'instant plus tranquille. Techniquement abandonné par Ashton Tate à la concurrence, n'a pas été réellement conquis par les superbes produits de Fox Software, le



Microsoft Visual Basic 1.0"

..... TELEX .....

➦ Le lancement de **MS DOS** aux USA peut se passer mal pour Microsoft. De son coté, IBM va présenter le 12 juin son ensemble de mise à jour, apportant quelques améliorations par rapport à la version Microsoft, notamment l'affichage en couleurs sur les moniteurs 8514 et meilleur support des Réseaux Locaux. En plus, l'ensemble IBM doit être moins chère : 49$ contre plus de 79$ de l'ensemble Microsoft.

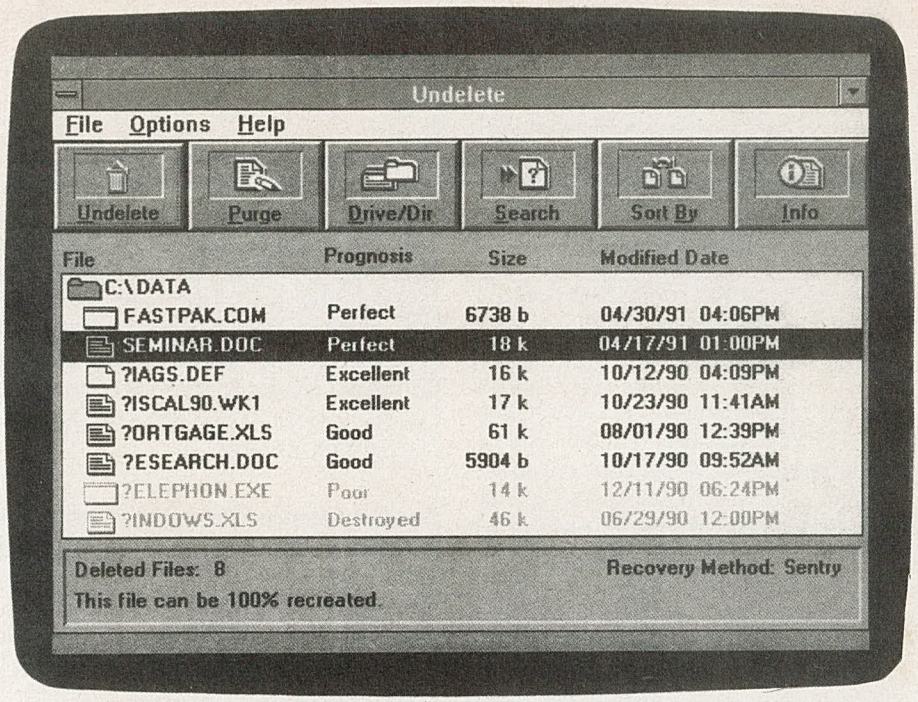
➦**Lotus** a choisi d'incorporer PostScript d'Adobe dans ses produits à venir. Ainsi, Lotus prend distance par rapport à True Type que Microsoft veut incorporer d'office dans Windows 3.1 à venir et qui fait partie intégrale de Système 7 livré actuellement par Apple.

➦ Les affaires sont excellentes pour les fabricants des disques durs miniaturisés, utilisés dans les portables. Conner Peripherials Inc. annonce la livraison de son millionième disque de taille 2.5" et cinq nouveaux modèles, parmi lesquels un disque de 85 Mo à 19 milisecondes, dont l'hauteur n'est que de 2 cm. Micropolis annonce aussi un nouveau modèle 2112 de 3.5" SCSI-2 dont la capacité après le formatage atteint 1 Go et la vitesse 10 ms! le prix en quantité sera environ de 1000 $ US.

➦ Il suffit de choisir le bon moment pour devenir riche. En février, Rod Canion, président de COMPAQ, a vendu 85% de son portefeuille d'actions détenus dans la compagnie, au prix de 9.8 millions de dollars. Deux mois plus tard, après l'annonce des mauvais résultats de COMPAQ, ce même paquet d'action ne vaut plus « que » 7.1 millions. La différence : 2 millions de $ de gain.

➦ On annonce de très mauvais résultats chez Businessland Inc., la plus grande chaîne de distribution à valeur ajouté en USA. Un dépôt de bilan est possible.

➦**INFOGRAF,** connu notamment pour lancer Novell et Banyan en France, vient de déposer le bilan. On parle d'un possible rachat de l'entreprise.



PC Tools 7 : nouveau UNDELETE pour DOS et Windows

marché semble dormir. Ashton Tate se remets des émotions récentes, ou le juge Hatter l'a dépourvu de ses droits de copie sur dBase. Le 23 avril le juge est revenu sur sa décision. Sur la demande d'Ashton Tate paniqué, le Bureau de Brevets américain a confirmé que l'enregistrement des droits a été effectué dans les règles et qu'il n'y a pas matière à contestation. Cependant, le jugement définitif au fond défavorable à Fox est devenu moins probable et Bill Lyons, président d'AT, est obligé maintenant à prendre en considération « un éventuel arrangement commercial du problème ».

```
(* PROGRAMME DE DÉMONSTRATION     *)
(* Turbo Pascal pour Windows      *)
(* par Serge Grun pour DEBUG MAGAZINE  *)

Program WinDemo;

  Uses WObjects;

  Type tWinDemoApp=Object(tApplication)
                     Procedure InitMainWindow; virtual;
                   End;

  Var  DemoApp:tWinDemoApp;

  Procedure tWinDemoApp.InitMainWindow;

    Begin
      MainWindow:=New(pWindow,Init(nil,'Bonjour, je viens d'arriver'))
    End;

  Begin
    DemoApp.Init('Demo');
    DemoApp.Run;
    DemoApp.Done
  End.
```

Ashton Tate travaille sur l'amélioration des performances, SQL et promet une version WINDOWS de la dBase IV. De son coté, Fox termine la version 2 de FoxPRO, qui va apporter des innovations technologiques majeures comme Rushmore, un nouvel optimiseur de requêtes, un compilateur et une API. Mais le coup décisif peut venir de Borland qui promet un compilateur compatible xBase (toutes incarnations) et Paradox Engine sous WINDOWS, pour la fin de l'année, connu sous l'appellation Turbo xBase.

Le champ suivant est celui de langages. Encore des géants : Borland et Microsoft, explosent littéralement la situation qui s'enlisait dans la lourdeur et la complexité. Visual Basic de Microsoft (disponible fin juillet en anglais au prix de 2 360 Frs en France et 199 $ US en Etats Unis) et PASCAL pour Windows de Borland (disponible actuellement en français pour 2 960 Frs ou 1 180 Frs en mise à jour) rendent à PC ses outils génériques. C'est par ses outils de programmation, faciles à utiliser et abordables, qu'un Ordinateur Personnel, par définition programmable par l'utilisateur, se différencie d'une Mainframe ou d'un MacIntosh. C'est le manque des langages de programmation, faciles à utiliser, qui était un de principaux obstacles à la popularité de Windows. Maintenant, compte tenu de la disponibilité de deux langages les plus courants on peut prédire l'explosion des applications Windows dans les mois à

.....TELEX.....

➦ **AMD** annonce les nouveaux processeurs 386SX et 386SXL. Les livraisons en quantité sont prévus pour juillet, le prix doit être comparable à celui des produits d'Intel.

➦ Une centaine d'avocats a été engagé par FTC pour mener l'enquête antitrust contre Microsoft. Ils sont à la recherche de toute information mettant en évidence les liens entre les départements OS et application chez Microsoft.

➦ **DISTRIBUTEUR RECHERCHE.** Gazelle Systems (éditeur de logiciels OPTune et Back-It 4) recherche un distributeur français pour traduire, localiser et diffuser ses produits. Contactez Mr Eugene L Seeley, Inal Sales Manager, 42 North University Avenue Suite 100 Provo, Utah 84601 USA téléphone (801) 377 1288 fax (801) 373 6933.

➦ **EDUCATION POUR LES HANDICAPES.** Association HELENE annonce l'ouverture de ses locaux - 20 place Michelet. 78800 HOUILLES. Tél : 39 57 90 90. C'est un projet d'éducation par l'informatique pour les enfants handicapés moteur (et d'autres, dans la mesure des possibilités). Les adultes peuvent aussi profiter des locaux équipés en PC et Mac, outils PAO et une large bibliothèque des logiciels.

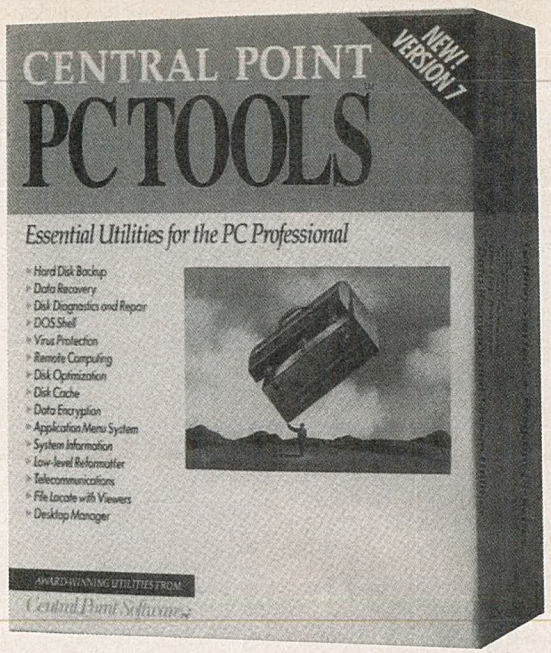
➦ **CAPITAL** - risque disponible. VP Partners recherche des projets ou entreprises européennes qui se trouvent dans un stade initial de développement ou des situations catastrophiques, mais ayant un potentiel. Contactez : M. Sandy Schupper, Venom Partners - 1 000 East Wiliam - Str. Suite 100 Carson City NV 89 701 - USA. Tél : (702) 885 2 588.

➦ **OFFSHORE Technologies** (15 av. Victor Hugo - 75116 Paris) recherche logiciels pour distribuer en Europe de l'Est. Contactez : M. Jules Zacher au (1) 44 28 19 49.

➦ **Qualitas** (éditeur de MAX386) recherche revendeurs. Contactez : Tod Currie ou Robert Hellemans chez Qualitas, avenue Lloyd George 6 - B 2 050 Bruxelles, Belgique au 19 (32) 2 648 5 470.

venir. Bien que probablement des programmeurs « industriels » ne vont pas abandonner C et SDK Win pour leur démarche « lourde » de fond, Visual Basic et Pascal Win vont permettre de programmer de petites applications « sur mesure » ou des maquettes fonctionnantes. D'autres utilisateurs, programmeurs occasionnels, consultants, vont pouvoir développer rapidement des solutions ponctuelles pour des clients particuliers.

Des simples utilisateurs enfin, vont pouvoir créer une multitude d'extraordinaires applications graphiques, pratiques ou de divertissement, surpassant de loin tout ce qui existe dans le commerce. Jugez vous même la facilité de programmation sous PASCAL pour WINDOWS (le listing manquant à l'article sur PASCAL pour WINDOWS paru dans le numéro 12), qui suit cet article.



PC Tools : La nouvelle version 7.0

Le cinquième champ de bataille est celui des utilitaires. Il commence progressivement de basculer aussi dans l'escarcelle Windows. Central Point Software annonce PC TOOLS 7.0. compatible DOS, Windows et Réseaux Locaux. La nouvelle version apporte « douze nouvelles fonctionnalités et plus de quatre-vingt améliorations ». L'ensemble inclut Central Point Backup 7.0 (pour DOS et pour Windows) et le nouveau logiciel Central Point Commute, qui permet de prendre en charge une autre PC à distance. Ce qui est nouveau par rapport à des utilitaires concurrents, c'est la possibilité de lancer un programme sur une station distante en Réseau, par exemple celui d'une copine qui est partie déjeuner. On peut même manipuler la souris à distance... Parmi d'autres nouveautés il y a la protection contre les virus et le formatage de bas niveau des disques durs. La mise à jour coûte aux USA, 49 $, CPS propose simultanément l'achat couplé d'un CP Antivirus à 30 $, plus les frais d'envoi de 15$. Total 94 $ (quelque 560 Frs). Les nouveaux acheteurs (ou les utilisateurs des produits concurrents) ont la possibilité d'acheter PCT 7 seul au prix de 59$+15$, 435 Frs, une affaire! Le prix définitif est fixé à 179$, 1060 Frs. En France, il va certainement franchir la barre de 2 000 Frs, la date de livraison n'est pas encore annoncée.

Symantec et Fifth Generation (respectivement Norton et Fastback) proposent aussi des programmes de « mise à jour » contre les produits concurrents à 59$, mais désormais personne ne peut rivaliser face à l'ensemble de fonctionnalités et qualité de PC TOLLS.

Un sixième champ qui « bouge » est celui des processeurs. Intel se plaint que « L'industrie électronique est caractérisée par des règles très spéciales. Ainsi, il apparaît moins grave de reproduire un processeur 386 que de broder un petit crocodile sur une chemisette polo. Dans les écoles primaires de la France profonde, il n'est pas rare d'entendre un gamin s'écrier « copieur! » à l'encontre d'un camarade qui, n'ayant pas travaillé sa leçon, essaie de lui subtiliser des bonnes réponses. Dans notre industrie, on s'exprime avec plus de distinction : Prenez temps, vous laissez votre camarade dessiner, déboguer, affiner, produire un processeur et vous attendez qu'un marché potentiel énorme apparaisse. Deuxième temps, vous vous levez d'un bond, vous expliquez qu'il « faut diversifier des sources d'approvisionnement » et qu'à cet effet vous avez décidé de vous livrer à une opération salutaire de « reverse engineering ». L'effet est assuré. Dans le même temps on écrase des milliers de fausses montres Cartier et les cloneurs de Lambada sont condamnés à reverser leur droits aux véritables compositeurs. « Tout cela suite à l'article de notre confrère 01 Informatique, qui titrait sur le « monopole d'Intel en matière de microprocesseurs ». Car, selon Intel, il n'y a aucun problème. Les puces sont en stock, il n'y a aucune pénurie. Plus loin, cette phrase sibylline : « Compte tenu de la complexité croissante des architectures, nous avons choisi depuis 1985 de gérer nos secondes sources en interne ». Tout cela a permis de vendre, quand même, 10 millions des i386 entre 1985 et 1990 et un million par mois depuis. Les résultats d'Intel s'en réjouissent : prés de 4 milliards de dollars de chiffre d'affaires en 1990 et 650 millions de bénéfice net. Une autre pensée innocente : « Par ailleurs, nous avons, sans concurrence, diminué le prix des i386 de 25% depuis cinq ans ». Et les économies d'échelle, on les oublie ? Je suis persuadé que depuis qu'AMD livre ses clones (« copieurs ! » crie t'on chez Intel), les baisses de prix vont être plus significatives et les chaînes vont tourner plus vite. Paradoxalement, cela ne va pas forcément diminuer les bénéfices d'Intel.

Nous ne sommes pas en France profonde, bien qu'on l'aime bien. Intel devrait se battre sur la plate-forme technologique et non juridique ou maladroitement commerciale. L'opération i486SX, processeur « dégradé » et dépourvu de l'unité arithmétique, est l'exemple même d'une chance perdue. Actuellement Intel est le seul à pouvoir intégrer un CPU et l'unité arithmétique dans une seule puce. C'est ce qui a été précisément fait dans le i486DX. Ainsi, un nouveau standard est né. De point de vue de programmeur et d'utilisateur, la puce ainsi crée a soudainement ajouté quelque quatre-vingt instructions supplémentaires à cent cinquante instructions de base de processeur de type antérieur (186, 286 ou

386). On pouvait légitimement s'attendre à ce que cela crée un nouveau jeu d'instructions complet et standard. Il faudrait longtemps pour que les concurrents d'Intel atteignent une telle plus-value. Malheureusement, pour des raisons de facilité commerciale, Intel est revenu sur sa décision et livre actuellement des i486SX dépourvus de l'unité arithmétique, qu'on n'arrive pas à situer. Doit-on le voir comme un 386 amélioré (alors pourquoi l'appeler 486?) ou comme un 486 dégradé (mais il ne l'est pas car en plus de baisse de vitesse les instructions manquent). Un micmac ou tout le monde est perdu. Et Intel sait mieux faire. L'exemple est le processeur i386SL, crée spécialement pour les portables.

Les portables sont le septième et énorme champ de bataille. Une véritable folie s'empare des clients : chacun veut « son » portable. Des vrais, légers, d'un poids inférieur à 3 Kg, non des monstres lourdauds et lents de 7 Kg ou plus. Le portable moderne, équipé de préférence d'un i386SX, a un écran VGA n/b, 2 Mo de mémoire, une disquette 1.44 Mo et un disque dur de 40 Mo. Ce type de machine coûte actuellement environ 3 à 4 mille dollars US, si vous voulez une étiquette d'une marque, vous allez payer davantage. Ce qui flanche, c'est l'autonomie, qui ne dépasse guère plus de 3 heures - « le temps d'un vol continental moyen » comme si tout le monde survolait quotidiennement le continent américain. C'est peu, surtout que les portables servent de plus en plus comme la machine principale : on le connecte au bureau sur un écran couleur et éventuellement, sur le réseau local, le soir on les emporte à la maison. On voyage avec, on passe des week-ends ensemble. Ces machines prennent de plus en plus d'importance et c'est dans ce segment que la concurrence entre quelque 150 marques actuelles est la plus rude. C'est aussi le marché ou les fabricants japonais risquent d'être les plus dangereux : pour produire une machine miniaturisée il n'y a qu'en Extrême Orient qu'on trouve des capacités industrielles adaptées, plus question d'assembler « sa » marque dans un coin. Une récente étude de JEIDA (Japan Electronic Industry Developement Association) prévoit même que les portables vont dépasser 50% de total de toutes les ventes des ordinateurs en Japon en 1991. En USA, la pénétration des portables est plus faible, selon IDC ils vont totaliser 13.3% de livraisons en 1991. Dans cette bataille, l'invention a beaucoup à dire : c'est justement le cas d'Intel et son jeu de circuits autour d'i386SL « frugal ». Ils intègrent la plupart des composants d'un AT et sachent gérer l'énergie de manière intelligente (par exemple, le processeur fonctionne à vitesse variable de 0 à 20 MHz). Il est possible désormais de construire un portable, ayant les capacités d'une machine de bureau performante, sur une plaquette de 10 cm sur 15 cm. La première réalisation pratique est française (un cocorico s.v.p.). C'est Zenith (Bull), qui vient de présenter justement une nouvelle ligne de portables, dont le modèle MasterPORT 386SL est construit sur cette base économique. Radine en énergie mais démesurément chère : la machine va coûter 45.000 Frs, avec le disque 60 Mo. C'est beaucoup payé pour faire des économies!

Il y a encore, certainement, beaucoup à dire sur tous les efforts déployés par notre industrie. Beaucoup de champs ou on s'emploie à être plus compétitif, avec les meilleurs produits ou meilleure approche du client. Il y a malheureusement un champ, ou on ne pourra parler que d'un échec, bien que certains crient pour l'instant à la victoire. C'est à Bruxelles que le (nomen omen) 13 décembre 1990 le Conseil de Communautés européennes a adopté la proposition commune en vue de l'adoption d'une directive concernant la protection juridique des programmes d'ordinateurs. La directive (n° 10562/1/90), dont certains articles sont imprécis et même contradictoires, est une victoire de lobby pro-américain SAGE bien que certains amendements sur décompilation de l'association européenne ECIS ont été adoptés. Elle protège d'une manière extrêmement rigoureuse les droits d'auteur, rendant tout écart ou « regard » illégal. Quand on sait que 85% de logiciels utilisés dans la CEE est d'origine étrangère (américaine), on comprend bien qui va être le principal bénéficiaire. Ce même type de protection n'existe pas nulle part ailleurs, ce qui va résulter en l'utilisation libérale des programmes européens dans le monde entier, sauf dans la CEE. Les européens, par contre, ne vont pouvoir disposer de la même manière des programmes étrangers. Seuls contents actuellement sont les SSII, mais leur joie risque d'être à court terme.

PRODUITS CITES

MS DOS 5.0
Visual Basic
Windows
SDK Windows
Excell
Microsoft (1) 69864646

DR DOS 5.0
Multiuser DOS
Digital Research (1) 39463366

Système 7
Apple 3614 code Apple

1-2-3
Lotus (1) 30458300

Pascal Windows
Quattro 3
Borland 39469669

FoxPro
Fox Software 19 1 (419) 874 0162
ou chez AB Soft (nouvelle adresse)
(1) 69337000

PC Tools 7
Central Point Backup
Central Point Antivirus
Central Point Software 19 1 (503) 690 8090.

Norton Outils 5
SYMANTEC 19 1 (408) 253 9600
ou chez FRAME (1) 47727777

Fastback
Fifth Systems Generation
19 1 (504) 291 7221

386SL
Intel (1) 30577000

MasterPort 386SL
ZENITH (1) 47781603

...Suite p. 75

Un Noyau Multi-tâches en C++

Un environnement multi-tâches, C++ qui laisse définir les tâches en tant qu'objets.

par *Tom Green*
traduit par Philippe Dalléas

Il y a environ deux ans que, dans la compagnie où je travaille, j'ai commencé à me servir d'un noyau multi-tâches sur notre Multibus 11 SCSI et les contrôleurs de communication. Cela a été ma première rencontre avec un noyau multi-tâches, et je me suis intéressé aux algorithmes nécessaires à l'exécution de multiples séquences de code avec un seul processeur. A titre expérimental, je décidai de tenter l'écriture d'un petit noyau (basculant d'une tâche à l'autre au rythme d'un compte-temps) destiné à l'environnement MS-DOS. Pour ma première tentative, au moyen de Turbo C, j'ai réalisé un petit noyau opérationnel en deux semaines. Il détourne de MS-DOS l'interruption du compteur de temps et change de tâche à chaque top du compte-temps (les tâches sont des fonctions C bouclant indéfiniment).

Quand le compilateur Zortech C++ sortit pour les machines MS-DOS, je l'ai acheté immédiatement et j'entrepris de récrire mon noyau. Les objets C++, à mon sens, sont parfaits pour un noyau multi- tâches. Les tâches sont des objets avec lesquels l'objet les gérant communique et qu'il commute. C++ peut aussi permettre aux objets-tâches de rendre toutes leurs données privatives, et les objets-tâches peuvent être créés au moyen d'un constructeur nanti de paramètres. L'objet gérant les tâches peut aussi garder privatives toutes les données et fournir des routines telles que les utilisateurs puissent ajouter des tâches, lancer et arrêter l'exécuteur multi-tâches, etc.

J'ai employé Modula-2 pour modèle en concevant les communications entre tâches. L'objet-signal est calqué sur le type données que l'on voit dans bien des progiciels Modula-2. Cet objet est employé par une tâche pour émettre ou recevoir un message, et il fournit une interface rapide et concise. J'ai ajouté une procédure d'arrêt (pour obliger une tâche à rendre la main) et une procédure de gestion de tâches pour qu'une tâche puisse inhiber la commutation en cas d'appel d'une procédure non réentrante (procédures MS-DOS, BIOS ou de la bibliothèque C).

Le noyau présenté ici peut sembler très succinct, mais c'est intentionnel. Si vous comprenez comment le noyau fonctionne, vous trouverez dès lors qu'il est facile à étendre et personnaliser. Certes y a-t-il de nombreuses stratégies pour concevoir un noyau, mais je pense que celle-ci fournit ce dont vous avez besoin pour la plupart des situations de programmation.

Aperçu sur les objets et la commutation de tâches.

Ce noyau emploie trois classes C++ : *task_control, task,* et *signal.* Si vous êtes novice en C++, voici une brève explication.

Une « classe » est semblable à une « structure » C ; en fait, en C++, une structure est une classe dont tous les éléments sont publics. Les données d'une classe peuvent être publiques ou privées. On peut associer à une classe des « méthodes » (synonyme de « fonction »). Un objet est une variable d'une classe spécifique. Quand un objet est activé, la place est réservée pour toutes les données (publiques et privées), et un jeu de méthodes (fonctions) agissant sur cet objet est aussitôt disponible. Ces méthodes sont employées exactement comme les données dans une structure, avec l'opérateur "." (point [zone choisie de la structure]) ou avec l'opérateur "->" (flèche [zone choisie de la structure et indirection]. Cela ne veut pas dire qu'il y ait des copies locales du code de la méthode pour chaque objet. Le compilateur Zortech (et aussi, je présume, les préprocesseurs de C++) passe à la méthode un pointeur désignant l'objet. C'est par ce pointeur que la méthode peut accéder aux données de l'objet.

A propos d'objets, je dois aussi faire référence aux « constructeurs » et « destructeurs ». Un constructeur est appelé quand un objet est déclaré; il sert à initialiser l'objet. Un destructeur est appelé quand un objet n'a plus d'utilité; il sert, la plupart du temps, à libérer la mémoire réservée par le constructeur.

Toutes les données de chaque objet du noyau sont privatives, afin d'empêcher quiconque de détruire accidentellement les opérations du noyau. Quelques procédures publiques (ou des méthodes) sont fournies pour l'ini-

tialisation et pour communiquer avec l'objet de gestion de tâches.

Un objet-signal est une file d'objets-tâches. Les données constitutives d'un objet-signal sont deux pointeurs relatifs à l'objet-tâche, qui servent à créer une file (pointeur de début et pointeur de fin).

C'est le plus simple objet dans le noyau. Il a seulement deux méthodes: *get_task_q,* qui prend le pointeur du prochain objet-tâche dans la file, et *put_task_q,* qui ajoute à la file un pointeur d'objet- tâche. Ces méthodes sont privées et ne peuvent être appelées que par l'objet *task_control* (lequel collabore avec l'objet- signal). Un constructeur d'objets « signal »initialise les pointeurs de début et de fin de file. Un objet « tâche » contient plusieurs lots de données employés par l'objet « gestion-de-tâche ». Hormis le « constructeur » et le « destructeur », il n'y a pas d'autre « méthode » agissant sur un objet « tâche ». Les objets « signal » et l'objet « gestion-de-tâche » sont des compagnons des objets « tâche ». Cela signifie qu'ils peuvent directement manipuler les données privatives d'un objet « tâche ». J'ai fait cela pour éviter les impedimenta que j'ai pu rencontrer en usant de méthodes agissant sur les données privatives d'un objet « tâche » C++ supporte les fonctions intégrées, mais, malheureusement, le compilateur Zortech ne produit pas le code intégré dans tous cas. Quand un objet « tâche »est déclaré, le constructeur est appelé et les données privatives initialisées. Une zone de travail, ou une pile, est réservée à l'objet « tâche » Une « image » de la tâche est enregistrée dans la mémoire réservée. L'« image » est une région de la mémoire conservant le contenu de tous des registres (8086) d'une tâche lors de son dernier arrêt. C'est pourquoi il faut que nous disposions d'une image des registres à utililer dès la première commutation sur la tâche. L'objet « tâche » comporte le pointeur de l'image de la tâche, lequel est chargé sur la pile.
Examinez la structure appelée *task_image* dans le listing 1, (task.hpp) afin de vous faire une idée de ce que à quoi ressemble l'image d'une tâche. On y voit ce qu'est devenue la pile après qu'un gestionnaire d'interruption ait été activé et que tous les registres aient été sauvegardés : c'est ainsi que le noyau enregistre l'image d'une tâche active avant de basculer vers une autre tâche.

Quand l'image d'une tâche est établie, la procédure getCS est appelée pour récupérer le segment de code de la tâche. C'est indispensable du fait d'une erreur du compilateur, laquelle renvoie le contenu du registre DS quand on veut obtenir le segment de l'adresse d'une fonction « near ». C'est à cause de cela que le noyau ne fonctionne que s'il a été compilé selon le modèle « small ».

L'objet-tâche comporte un indicateur d'état de la tâche et un pointeur d'objet-tâche. Ce pointeur, *next_task* sert aux méthodes *signal* pour ajouter à la file *signal* des objets-tâches, ou les en retirer. Le destructeur d'un objet-tâche libère la mémoire réservée à la pile d'objets-tâches, ou espace de travail. L'objet *task_control* s'occupe de commuter les tâches et fournit les moyens de communication avec l'extérieur. Ses méthodes permettent à une tâche de capter ou émettre des signaux, inhiber ou non la commutation de tâches, etc ... Pour que le multi-tâches soit actif, il faut avoir donné à l'objet *task_control* des objets-tâches via *add_new_task.* L'objet *task_control* possède un objet- signal, dénommé *ready_q* auquel sont déclarées toutes les tâches en instance. C'est objet ne sert pas en tant qu'aiguillage, mais en tant que file d'attente. Lorsque tous les objets-tâches ont été déclarés, l'appel à *start_tasks* sauvegarde le gestionnaire initial des tops d'horloge, le remplace par un autre et lance la première tâche dès que survient le premier top d'horloge. L'objet *task_control* commute les tâches jusqu'à ce que *stop_tasks* soit appelé. Quand cela advient, le gestionnaire originel des tops d'horloge est rétabli, et le fonctionnement reprend au point suivant l'instant où la procédure *start_tasks* avait été appelée.

Quand le noyau fonctionne, les tâches sont commutées 18, 2 fois par seconde. Lorsque survient un top d'horloge, la tâche active est ajoutée à *ready_q* et la prochaine tâche à activer y est prélevée. Elle fonctionne jusqu'au prochain top d'horloge, et ainsi de suite.

Les tâches peuvent aussi s'interrompre d'elles-mêmes et appeler le noyau pour qu'il active la tâche suivante dans *ready_q*. Une tâche peut bloquer, attendre ou émettre un signal. Se servir de *send* et de *wait* permet aux tâches de communiquer.

Une tâche peut utiliser *task_control::block()*, qui n'a pas besoin de paramètre, pour rendre volontairement la main. La tâche appelante est ajoutée à *ready_q* et la tâche suivante y est prélevée. C'est utile pour que d'autres tâches soient activées lorsque l'une vient d'être accomplie.

Une tâche appelle *task_control::wait(),* avec, pour paramètre, l'adresse d'un objet-signal, pour attendre un signal en provenance d'une autre tâche. La tâche appelante est ajoutée à la file des objets-signaux et la prochaine tâche à activer est prélevée de la file *ready_q.*

Les tâches appellent *task_control ::send(),* avec, pour paramètre, l'adresse d'un objet-signal, pour envoyer un signal à une tâche en attendant un. S'il y a des objets-tâches dans la file d'objets- signaux, la tâche se trouvant en tête de file en est prélevée, puis elle est ajoutée à la file *ready_q.* L'objet-tâche appelant est à son tour ajouté à *ready_q.* La tâche suivante à activer est prélever en tête de *ready_q.*

L'objet *task_control* est pourvu de deux méthodes pour activer et désactiver la commutation de tâches par le top d'horloge. Cela permet à une tâche d'empêcher la commutation lorsqu'elle appelle une procédure non réentrante (MS-DOS, BIOS ou bibliothèque C). On appelle *task_control::lock()* pour inhiber la commutation, et *task_control::unlock()* pour la rétablir. Ces procédures ne concernent pas une tâche en train de rendre la main et d'appeler le noyau (*block, send* ou *wait).*

Une mise en garde à propos de l'objet *task_control :* il ne peut y en avoir qu'un. En considérant le listing 2, vous apercevrez une variable globale appelée *gl_tptr,* qui pointe sur un objet *task_control.* Dans le constructeur d'un objet *task_control*, ce pointeur est chargé avec l'adresse de l'objet *task_control.* Le pointeur *gl_tptr* est utilisé par le gestionnaire des tops d'horloge et la procédure *save_image* dans le listing 3 (timer.asm). C'est ce pointeur implicite (passé à une procédure C++) qui est indispensable pour appeler *task_control::task_switch().* Ce n'est pas très élégant, mais cela permet d'écrire en C++ le code de la commutation de tâches, et à *task_switch* d'accéder aux données privatives des objets-tâches et des objets-signaux (en en devenant compagnon).

■

Détails de la commutation de tâches.

La commutation de tâches n'est pas possible si l'on ne conserve pas l'image d'une tâche quant au contenu de la totalité des registres 8086. Cette sauvegarde est faite par les procédures *timer_handler* et *save_image* du listing 3. Elles font toutes deux la même chose : elles enregistrent l'image de la tâche en déposant sur la pile des tâches les registres indispensables, puis appellent la procédure en C++ *task_switch*. Regardons de plus près ces procédures.

Le nouveau gestionnaire des tops d'horloge se nomme *timer_handler.* Il sauvegarde tous les registres indispensables et recharge le registre DS afin que le code C++ puisse y trouver ses données. Le gestionnaire originel des tops d'horloge est invoqué pour prendre en compte les besoins chronométriques de la machine. Ensuite, différents paramètres sont déposés en prévision de l'appel à la procédure en C++ *task_switch*. J'analyserai plus loin ces paramètres. Après appel à la procédure *task_switch,* le pointeur « far » de l'image du nouvel objet-tâche est renvoyé. Les registres SP et SS sont chargés de ce pointeur et les registres du nouvel objet-tâche sont extraits de la pile. Une instruction iret est exécutée, et nous nous retrouvons dans la nouvelle tâche.

La procédure en assembleur *save_image* est appelée par les méthodes en C++ *block, send* et *wait.* Elle sert à sauvegarder l'image d'une tâche exactement comme le fait *timer_handler.* Cette procédure est un peu « rusée » : deux paramètres sont déposés sur la pile (un indicateur et un pointeur de signal) ; ils en sont extraits pour être passés à la procédure en C++ *task_switch,* puis empilés à nouveau ; enfin, la pile est réorganisée afin que les indicateurs et une adresse « far » de retour soient sur le dessus (comme s'il y avait eu interruption) ; les registres inutilisés sont empilés, comme dans *timer_handler.* La procédure s'achève exactement comme celles de *timer_handler;* la méthode en C++ *task_switch* est invoquée, puis, au retour, les registres sont chargés afin que la tâche suivante soit exécutée.

La méthode *task_switch* est appelée par les procédures *timer_handler* et *save_image*. Elle passe le pointeur « far » de l'image de la tâche active. Le pointeur de l'objet-tâche est ajouté à *ready_q* ou à la file d'un signal. L'indicateur passé par les procédures sert à informer *task_switch* de la provenance de son appel. *task_switch* a besoin de savoir s'il s'agit d'un top d'horloge, ou d'un *send, wait* ou *block.* Une fois que *task_switch* sait quelle tâche activer, le pointeur « far » de l'image de cette tâche est rendu à *save_image* ou à *timer_handler.* La tâche devient active dès que les registres sont rechargés et qu'un iret est exécuté.

J'espère que tout cela vous a paru aussi simple qu'il en est réellement de commuter des tâches. Dans la commutation de tâches, l'important est de sauvegarder tous les registres de la tâche active puis de sauvegarder le pointeur « far » dirigé sur cette portion de la mémoire. Chaque tâche a sa propre pile (mémoire attribuée au moyen de *malloc*). Le pointeur « far » vise dans cette pile attribuée l'endroit où ont été déposés les registres. Remettre une tâche dans l'état où elle se trouvait, lors de sa dernière activité, se fait par le simple recueil d'un pointeur « far », son chargement dans les registres SS et SP, la récupération des registres et l'exécution d'un iret. La nouvelle tâche est alors active.

Un programme de démonstration multi-tâches

Pour compiler le programme multitâches, objet du listing 4, et qui met en évidence les concepts exposés ci-avant, tapez *ztc taskdemo task timer.* Le programme *ztc* de Zortech prend en compte chaque fichier source et met en oeuvre le compilateur C++ ou l'assembleur, selon chaque cas. L'éditeur de liens crée ensuite le fichier exécutable.

Le programme *taskdemo* est constitué de cinq tâches mettant en évidence les différents moyens par lesquels commuter des tâches. Rien d'autre n'intervient dans le programme. Les cinq tâches se déroulent et des compteurs tournent. La tâche *task0* s'occupe d'afficher les compteurs des autres tâches. Examinons chacune d'elles.

La tâche *task0* fait un appel à *lock()* pour qu'il n'y ait pas de commutation tant qu'elle n'a pas tout envoyé à l'écran. Le compteur propre à chacune des quatre autres tâches est affiché. Ensuite, *stop_tasks()* est appelé si le clavier a été touché. Si ce n'est pas le cas, la procédure *unlock()* est alors appelée pour réactiver la commutation de tâches. Enfin, la procédure block() est appelée pour permettre aux autres tâches de fonctionner.

Les tâches *task1* et *task2* font tourner leurs compteurs jusqu'à la survenue d'un top d'horloge. Elles disposent d'un grand laps de temps, d'où l'accroisement très rapide de leurs compteurs.

La tâche *task3* incrémente son compteur puis appelle *wait(),* qui permet à la tâche suivante de s'activer. Après quoi, *task4* incrémente son compteur et appelle *send,* ce qui a pour effet de mettre *task3* dans la file *ready_q.* L'appel à send permet d'activer la prochaine tâche se trouvant dans la file *ready_q.* L'effet visible en est que les compteurs de *task3* et *task4* tournent lentement de par le fait que chaque tâche n'en active une autre qu'après incrémentation de son compteur.

Stratégie d'un noyau multi_tâches

Il est aisé de modifier ou ajouter des propriétés à ce noyau élémentaire. Par exemple, dans les intégrations de send en Modula-2 que j'ai vues, s'il n'y a pas de tâche dans la pile signal, un drapeau est alors levé de façon à ce que la prochaine tâche, qui viendrait en attente dans la file *signal,* en soit avertie. Dans ma version du noyau, ce message n'est pas pris en compte. Si vous estimez qu'un message ne doit pas être perdu, il vous suffit d'ajouter un indicateur à la classe *signal.*

Une communication entre tâches à la Modula-2 constituait une excellente approche d'un noyau « simple et succinct ». Cela permet à l'objet *task_control* de s'informer via une très petite structure de données, car les objets-signaux comportent toutes les informations dont ils ont besoin. La relation avec ces objets-signaux consiste en une couple de méthodes simples et rapides prenant et posant des pointeurs dans une file. L'objet *task_control* se sert des mêmes méthodes simples pour prélever dans *ready_q* la prochaine tâche à activer.

...Suite p. 75

Un noyau multi-tâches fondé sur le principe de « préséance »

par *Allen Holub*
traduit par Philippe Dalléas

Cet article porte principalement sur un petit noyau multi-tâches à « préséance » destiné aux PC (voir listings 1 à 7, page () et suivantes).
La plus grande partie du noyau est rédigée en langage C, aussi ne sera-t-il pas trop difficile de le porter dans un autre environnement, fût-il en ROM.
Le noyau permet de pratiquer l'exécution multi-tâches à l'intérieur d'un programme.
En fait, il vous fait exécuter des sous-procédures en tant que tâches indépendantes.
Il supporte aussi une messagerie réservée à la communication entre tâches (et coupe quand un message n'est pas reçu dans le temps imparti).
Il ne permet pas d'exécuter ensemble plusieurs programmes et il utilise le DOS en tant que système d'E/S.

Qu'est-ce que l'exécution multi-tâches ?

Commençons par quelques définitions, afin que chacun parte du même pied.

Un système d'exploitation est un ensemble de programmes permettant d'en exécuter d'autres. De façon typique, certains de ces programmes sont résidents (ils restent dans la mémoire de l'ordinateur), d'autres ne le sont pas (ils restent sur disque jusqu'à ce qu'on en ait besoin, auquel cas ils sont chargés en mémoire et exécutés) et quelques uns connaissent une situation intermédiaire (ils restent en mémoire jusqu'à ce qu'on ait besoin de place, auquel cas ils sont écrasés puis rechargés dès que la place sera redevenue disponible).

Les principales parties d'un système d'exploitation sont le noyau (ou exécuteur), le système d'E/S et la coque (« shell»). Le noyau a pour mission le chargement et l'exécution des autres programmes, le système d'E/S gère les communications avec le monde extérieur (disques, imprimantes, terminaux, etc ...) et la coque est ce avec quoi vous êtes en contact. En l'espèce, la coque se sert du système d'E/S et du noyau. Ces colonnes n'ont pas pour but de présenter un système d'exploitation, mais d'en examiner une parcelle de l'exécuteur, en l'occurrence celle qui gère l'exécution multi-tâches. L'exécution multi-tâches est un procédé faisant apparaître un ordinateur comme capable de faire plusieurs choses à la fois. En réalité, il va de l'une à l'autre très rapidement. Néanmoins, du point de vue humain, les choses semblent se passer simultanément. On peut actuellement faire du multi-tâches en l'absence d'un exécuteur. Considérons un multiplexeur de terminaux gérés par interruptions - un appareil recevant des lignes de texte de plusieurs terminaux et les passant (accompagnées de différentes sortes d'informations d'identité) via un canal de communication à un unique ordinateur principal. On disposerait de huit canaux d'entrée, chacun déclenchant sa propre interruption du processeur. Il faudrait alors huit procédures de gestion d'interruption, chacune recueillant les caractères jusqu'à rencontrer la marque de fin de ligne puis envoyant la ligne à l'ordinateur principal. Chaque procédure gérerait son propre tampon d'entrée et marquerait chaque ligne d'un élément unique d'identité. Le multiplexeur aurait l'air de faire huit choses à la fois, en recueillant des données sur huit canaux simultanément, mais, en réalité, il exécuterait huit tâches indépendantes sous la sollicitation d'événements externes (arrivages de caractères). Si huit caractères arrivaient ensemble, le multiplexeur serait pris en défaut, manquant de temps pour réagir aux huit entrées avant que d'autres caractères n'arrivent.

Une approche différente du même problème consiste à ne gérer qu'une seule interruption, ordinairement produite par tout genre de composant compte-temps, laquelle déclenche une procédure de gestion de file d'attente. Le gestionnaire de file d'attente active une sous-procédure particulière,

[SERVICE LECTEURS : INTERESSANT : 10 / MOYEN : 11 / MAUVAIS : 12]

TE_NOERR	Pas d'erreur
TE_TOOMANY	Le nombre maximal de tâches est atteint (32)
TE_NOMEM	Mémoire disponible insuffisante
TE_BADARG	Argument illicite
TE_TIMEOUT	Délai dépassé
TE_QFULL	La file d'attente est saturée
TE_NOTASKS	Pas de tâche à qui envoyer le message
TE_INTERNAL	Erreur interne
TE_DEADLOCK	Suppression entraînant une impasse
TE_STACK	Dépassement de capacité de pile
TE_KILL	CTRL-BREAK a été activé

***Table 1 :* Codes d'erreurs déclarés dans kernel.h**

comme si l'interruption le faisait directement elle-même. Au top du compte-temps, le gestionnaire de file d'attente arrête la sous-procédure en cours pour en activer une autre, exactement comme si une seconde interruption, de plus haute priorité, était intervenue. Le gestionnaire de file d'attente prend aussi en compte les préfixes impliqués dans la gestion d'interruption. En particulier, il effectue un changement de contexte - il sauvegarde tous les registres (y compris le pointeur d'instruction) et les piles des sous-procédures en instance (chaque tâche possède sa propre pile, tout comme une procédure de gestion d'interruption bien faite doit avoir sa propre pile). Une tâche, enfin, diffère d'une sous-procédure normale en ce qu'elle n'est activée que par le gestionnaire de file d'attente, au lieu de l'être par un classique appel à sous-procédure, et qu'elle possède sa propre pile et une zone où sauvegarder les registres au moment d'une commutation.

Notez qu'une tâche n'est pas la même chose qu'un processus, dans le sabir des systèmes d'exploitation. Ordinairement, un processus implique un programme chargé depuis le disque puis exécuté par le système d'exploitation. A l'inverse, une tâche est tout code indépendamment exécutable. Dans le cas de l'exécuteur présenté ici, une tâche est une sous-procédure (et les sous-procédures qu'elle appelle) possédant ses propres pile et jeu de registres. En l'occurrence, c'est la zone de sauvegarde de la pile et des registres qui définit une tâche, et non le code lui-même. Quelques tâches peuvent partager le même code, pourvu qu'il n'utilise pas de variables statiques. Parce que chaque tâche a sa propre pile et que le séquenceur de programme est sauvegardé en tant qu'élément de changement de contexte, les variables locales (situées en haut de pile) doivent être distinctes et chaque tâche doit savoir à quel endroit du code elle était rendue lors de son interruption. Toutefois, beaucoup de codes de changement de contexte assument que tout est contenu en un seul programme exécutable selon le modèle 8086 restreint. Vous pouvez à loisir modifier ce programme pour qu'il supporte d'autres modèles ou qu'il exécute des programmes externes.

```
TCB             *T_active;
unsigned long   T_clock;
int             T_numtasks;
T_QUEUE
*T_queues;
PQ              *T_tasks;
void            _t_reschedule();
void            _t_install();
void            _t_swap_in();
void            _t_shazam();
void            _t_speedup();
```

Table 2 : Variables et sous-procédures globales utilisées par kernel.h

Plusieurs voies sont suivies pour déterminer à quelle tâche donner privilège quand survient le top d'horloge. En gestion de files d'attente, les différentes tâches sont activées à tour de rôle, jusqu'à ce qu'elles l'aient toutes été, puis le processeur refait sa ronde. En termes de priorités, on en associe une à chaque tâche. Les tâches en instance sont rangées dans une liste d'actions. Quant une interruption survient, le gestionnaire compare le degré de priorité de la tâche active avec ceux des tâches en instance et, s'il le faut, met en sommeil la tâche actuelle au profit de la première de celles ayant, dans la liste d'attente, la plus haute priorité. Un système peut très bien utiliser ces deux stratégies, les tâches de même priorité étant animées chacune à son tour, tandis que celles de plus haute priorité prennent le pas sur celles de rangs inférieurs.

Il existe aussi deux autres méthodes de gestion multi-tâches, très utiles dans certaines situations. La méthode que je viens de décrire constitue une gestion multi-tâches sur « préséances ». Le top d'horloge incite la tâche suivante à exercer son « droit de préséance » sur l'actuelle. On peut très bien recourir à un système sans « droit de préséance » (comme Microsoft Windows), dans lequel une tâche ne rend la main qu'accomplie. Il y a deux avantages à cette approche. On n'a pas besoin des tops d'horloge et on ne gaspille pas de temps de CPU à s'apercevoir que le gestionnaire de la file d'attente n'a nul besoin de procéder à un échange de contextes. En revers de médaille, une tâche garde la main jusqu'à ce qu'elle soit accomplie ; si elle se « plante », c'est la totalité du système qui est bloquée.

```
idle (\scl28\)
{
   int t, i ;
 do {
      for( i = 1000; —i>=O ; )
         ;
      t_cli (scl28\);  /* Clear interrupts */
      t = T_numtasks;  /* t = # of tasks */
      t_sti (\sc128\); /* restore ints */
   }
   while( t > 1 ) ;
 t_delete( NULL );   /* delete self */
}
```

Exemple 1 : Une tâche typique d'attente

Parce que les tâches ne sont pas des sous-procédures, au sens habituel du terme, elles n'échangent pas d'informations avec les autres par le biais d'arguments et de résultats. La communication entre tâches se réalise alors grâce à un système de messagerie. Le système d'exploitation permet de créer une « poste restante » ou un jeu de « boîtes postales », auxquelles les messages sont envoyés. Les autres tâches vont voir si un message leur est parvenu. Plusieurs tâches peuvent attendre ensemble chacune son message. S'il n'y a pas de tâche en attente, chaque message arrivant est emmagasiné jusqu'à ce qu'une tâche vienne le chercher. Une tâche en attente d'un message est suspendue - elle ne sera réactivée par le gestionnaire qu'après l'arrivée du message - mais elle peut être aussi hors délai - la tâche est alors remise dans la liste d'activités si le message n'a pu arriver dans le temps imparti. Un message étant arrivé, si la tâche destinataire en est de plus grande priorité que celle en cours, elle prend aussitôt la main; sinon, la tâche et le message sont mis dans la liste d'instances avant d'être activés normalement. Notez qu'une « boîte postale » est une structure de données contenant deux files : celle des messages et celle des tâches attendant un message.

L'inconvénient de cette méthode de communication, c'est l'impasse, situation où l'on se trouve quand deux tâches attendent un message que seule l'autre peut lui adresser. Une situation comparable advient lorsque toutes les tâches du système attendent simultanément un message.

Un autre inconvénient est le blocage de ressources. Le système d'E/S du DOS est un bon exemple de la nécessité du blocage de ressources. DOS n'est pas « réentrant », ce qui signifie qu'une fonction DOS ne peut pas être appelée en toute confiance depuis une procédure de gestion d'interruptions. En d'autres termes, quand vous venez de faire un appel à DOS, vous ne pouvez en faire un second que le premier parachevé. Parce que l'activation d'une tâche peut survenir à tout instant, même si vous vous trouver dans le DOS, il faut suivre une méthode particulière pour arrêter le gestionnaire quand une tâche veut faire appel au DOS. Une tâche bloque pour prendre contrôle du CPU - aucune autre ne peut être activée tant que le gestionnaire restera bloqué. Notez que le système s'arrête, quand le gestionnaire

```
foo ( a, b )
char *a, *b;
{
   t_printf("%s %s\n", a, b );
   t_delete( NULL );
}
main (\scl28\)
{
   t_create( foo, "foo", 10, 512, "hello", "world", NULL);
   t_start( 2);
}
```

Exemple 2 : Création d'une tâche

est bloqué, et que la tâche en cours est « plantée ».

Une autre méthode de blocage, n'ayant pas cet inconvénient, consiste en l'utilisation d'un sémaphore d'exclusion. Un tel sémaphore est équivalent à une file constituée d'un seul élément. Un message y est déposé quand une ressource (par exemple, DOS) est disponible. Pour utiliser la ressource, une tâche prend le message, occupe la ressource puis redépose le message pour libérer la ressource. Cette méthode

```
#include <stdio.h>
#include <tools/vldeo.h>
#include "kernel.h"

#define TEST 3 /* 1 = test "sans préséance",
                * 2 = test "à préséance" : simple compte-temps
                * 3 = test "à la queu leu leu"
                */

T_QUEUE *Queue1;
T_QUEUE *Queue2;

main (\scl28\)
{
   int status = O;
   long t_numint (\sc128\), t_numblk (\sc128\);
   int sam(\sc128\), dave(\sc128\), timer(\sc128\), idle(\sc128\), maintask;

if ( ! (Queue1 = (T_QUEUE *) t_makequeue ( 2 ) ) ) printf("Can't make Queue1
queue\n"), exit(1);

if ( ! (Queue2 = (T_QUEUE *) t_makequeue ( 2 ) ) ) printf("Can't make Queue2
queue\n"), exit(1);

#if (TEST == 1)
   status = (int) t_create(sam, "sam", 100, 512, "SAM", NULL);
   status = (int) t_create(dave, "dave", 50, 512, "DAVE", NULL);
   status = t_start( 0 );
#endif

#if (TEST == 2)
   status = (int) t_create(timer, "timer", 10, 512, "timer", NULL);
   status = (int) t_create(idle, "idle", 1, 100,     NULL);
   status = t_start( 2 );
#endif

#if (TEST == 3)
```

Exemple 3 : Programme créant des files d'attente, demarrant l'exécution multi-tâche et procédant à trois tests

```
dave ( arg )
char *arg;
char *s;
{
t_printf ! "In dave (%s), about to wait for messages\n", arg );

if ( t_iserr(s = t_wait (Queue1, 100) ) )
t_perror("dave: first wait call", (int) s );
else
t_printf ( "dave: got <%s> from Queue1\n", s );

if ( t_iserr (s = t_wait (Queuel, 100) ) )
t_perror("dave: first wait call", (int) s );
else
t_printf ( dave: got %s from Queue1\n", s );

t_printf ("dave: yielding\n");
t_yield (\sc128\);

t_prlntf("dave: deleting self\n");
```

Exemple 4 : La tâche dave()

a, certes, ses inconvénients. Si la tâche, qui a réservé la ressource, est suspendue pour quelque raison, aucune autre tâche ne peut accéder à la ressource tant que la première n'a pas été réactivée.

Notez que, bien qu'on puisse parfaitement procéder au blocage par inhibition des interruptions, ce n'est pas une bonne manière de faire. En tout premier lieu, sur un PC, on va perturber l'horloge du système gérée sur interruptions. Les autres ennuis porteront sur les sous-systèmes E/S de la console et du disque, eux aussi gérés sur interruptions. Vous serez incapable d'envoyer ou recevoir des caractères si les interruptions sont inactives. De manière comparable, vos transferts sur disque en seraient perdus. Si vous deviez inhiber les interruptions, faites-

```
sam ( arg )
char *arg;
{
int err;
t_printf( "In sam(%s), yielding: no messages\n", arg);
t_yield(\sc128\);
t_printf ( "sam: back from yield, sending messages\n");

if( err = t_send ( Queue1, "1st message") ) t_perror( "Foo: sending 1st
message", err ) ;
if( err = t_send( Queue1, "2nd message") ) t_perror( "Foo: sending 2nd
message", err ) ;

if( err = t_send( Queue1, "3rd message") ) t_perror( "Foo: sending 3rd
message", err ) ;
t_printf ("Foo:  yielding again\n");

t_yield (\scs128\);
t_printf("Foos Returned from 2nd yield, deleting self\n");
t_delete( NULL );
}
```

Exemple 5 : La tâche sam()

le pour la plus brève durée possible. Notez aussi que les tâches doivent être considérées comme des procédures de gestion d'interruptions. En pratique, cela veut dire que vous devez toujours faire une réservation avant d'envoyer un appel au DOS.

Manuel d'utilisation du noyau

Cette section n'est qu'un guide d'utilisation des sous-procédures du noyau. Je n'entrerai pas dans les détails d'installation, qui feront l'objet d'une prochaine publication.

Différents codes d'erreur sont déclarés dans *kernel.h* et sont renvoyés par les diverses sous-procédures de l'exécuteur multi-tâches. Ces codes apparaissent dans le tableau 1 ci-après. *TE_NOERR* vaut 0 et les autres codes sont des petits nombres négatifs. En plus, *kernel.h* contient *typedefs* pour *TCB* et *T_QUEUE.* Le premier est un bloc de contrôle de tâche, contenant la pile et autres éléments, et le second est une file de messagerie. J'approfondirai tous ces codes un peu plus tard.

Diverses variables globales et sous-procédures sont utilisées de manière interne. Elles sont inventoriées dans le tableau 2, ci-après. Vous n'êtes pas obligé d'utiliser leurs noms dans vos sous-programmes, parce que, dans la plupart des cas, ces variables n'ont aucune utilité dans un programme d'application, sauf, peut-être, *T_clock,* l'horloge du système. *T_clock* est accru par le tic-tac du système, sauf quand le gestionnaire est bloqué. Parce que *T_clock* est un « unsigned long », il n'y a pas de risque de dépassement. Si l'on considère qu'on a par défaut 18,2 battements par seconde, le compteur sera en dépassement au bout de 65.552 heures, soit environ 7,47 ans :

((0xffffffff/18,2)/60)/60=65.552 heures
65.552/25/365,35=7,47798 ans

Bien entendu, ce nombre varie en fonction de la fréquence des battements, mais la précision reste convenable pour des fréquences variant dans des limites raisonnables. L'autre variable pouvant avoir de l'intérêt est *T_numtasks,* contenant le nombre de tâches qui ont été créées.

Restent quelques observations complémentaires. Le compilateur Microsoft insère un appel à une sous-procédure,

```
In sam(SAM), yielding: no messages
In dave(DAVE), about to wait for message
sam: back from yield, sending messages
dave: got <1st message> from Queue1
dave: got 2nd message from Queue1
dave: yielding
Foo: yielding again
dave: deleting self
Foo: Returned from 2nd yield, deleting self
done: No error
Scheduler called 0 times: 0 tasks timed-out, 0 context swaps
0 interrupts, 0 blocked
```

Exemple 6 : Resultat du test 1 de l'exemple 3

dénommée *_chkstk(),* en début de toute sous-procédure. Cela peut poser problème du fait qu'une tâche agit sur sa propre pile et non sur celle que *_chkstk()* s'attend à voir utiliser. Le problème est résolu par *t_start(),* qui remplace la fonction par défaut *_chkstk()* par la sienne propre. La nouvelle fonction compare la valeur en cours du pointeur de pile avec la base des piles locales des tâches. S'il y a dépassement de pile, l'exécution multi-tâches est terminée et *t_start()* renvoie un code d'erreur (voir plus loin).

Démarrer et arrêter l'exécution multi-tâches

Un programme typique commence par la déclaration de quelques tâches (on verra plus loin comment) puis démarre l'environnement multi-tâches (ces tâches peuvent aussi elles-mêmes en engendrer d'autres). Il n'est pas indispensable qu'elles soient connues au démarrage du multi-tâches. L'exécution multi-tâches est lancée en invoquant :

int t_start(speedup)
int speedup

Speedup est un facteur d'accélération de l'horloge système. S'il est à 0, le système n'est pas « à préséances » et il faut que les tâches soient achevées (soit par appel à *t_yield(),* soit par attente d'un message) pour rendre la main. Un facteur d'accélération égal à 1 fait qu'on utilise la fréquence standard des PC (soient 18,2 interruptions par seconde). Avec un facteur égal à 2, on est deux fois plus rapide (36,4 interruptions par seconde). Si le facteur est trop élevé, le système va se ralentir du fait de la perte d'interruptions. Au mieux, le facteur d'accélération doit être une puissance de 2.

Une tâche, au moins, doit avoir été déclarée avant que d'appeler *t_start().* La main est aussitôt donnée à la tâche de plus haute priorité. *T_start()* s'arrête quand toutes les tâches sont terminées ou quand on a appelé *t_stop()* (analysée plus loin). *TE_NOTASKS* est renvoyé à l'état « vrai », s'il n'y a pas de tâche prédéclarée - auquel cas l'exécution multi-tâches n'est pas lancée. *TE_NOERR* est renvoyé à l'état « vrai », lorsque toutes les tâches se sont correctement achevées. Aucune tâche ne doit être en file d'attente. C'est la façon normale de finir. *TE_STACK* est renvoyé dès que, dans une tâche, est rencontré un dépassement de pile. *T_active* pointe sur le bloc de contrôle (TCB) de la tâche délictueuse.

TE_DEADLOCK est renvoyé quand la seule tâche active dans le système s'abolit d'elle-même. D'autres tâches existent, mais elles sont toutes en file d'attente. Cela signifie qu'il doit toujours y avoir au moins une tâche active dans le système. Pour y satisfaire, une tâche oisive à très basse priorité est souvent créée. Cette tâche n'a rien d'autre à faire qu'à tourner; elle n'est active que lorsque les autres tâches font autre chose, et elle s'autolyse quand elle devient la seule tâche du système. Une tâche oisive typique est rencontrée dans l'exemple 1, ci-après.

Si *TE_NOERR* est renvoyé, c'est que toute la mémoire attribuée aux tâches est récupérée; sinon, si l'une des erreurs précitées a été renvoyée, *T_active* pointe sur le bloc de contrôle de la tâche fautive. Il est possible que d'autre valeurs soient renvoyées si une tâche appelle *t_stop()* directement (voir plus loin).

Depuis l'environnement multi-tâches, un « arrêt d'urgence » peut être provoqué au moyen de :

t_stop(errcode)
int errcode;

Le multi-tâche est interrompu, et la main est rendue à la procédure qui avait appelé t_start() (juste après cet appel). En fait, *t_stop()* oblige *t_start()* - qui ne peut le faire de soi-même - à rendre la main. *Errcode* est renvoyé à la procédure appelante en tant que valeur résultante de *t_start().* Ce procédé est analogue à celui d'un appel à *exit()* sous Unix, qui ne revient pas, et dont l'argument est contre-passé à un appel *wait()* dans le processus « parent ».

■

```
timer (\scl28\)
{
int i;
t_printf ("Starting up timer\n");
for ( i = 5; —i >= 0; )
{
t_printf( "!" );
t_wait( Queue2,t_second(\sc128\) );
}
t_printf ("Deleting timer\n");
t_delete ( NULL );
}
#endif
```

Exemple 7 : La tâche timer()

Créer et supprimer des tâches

Les tâches sont créées au moyen de l'appel à :

*TCB*t_create(subr,tag,priority,stack_size,.....,NULL)*
*int (*subr)();*
*char *tag;*
unsigned priority;
int stack_size;

Subr pointe sur la sous-procédure constituant le module principal de la tâche. *Tag* est une chaîne servant à caractériser le *TCB* (bloc de contrôle de tâche). Cela ne sert qu'à « épucer »(1). *Priority* est le niveau de priorité de la tâche - plus la valeur en est élevée, plus la tâche est prioritaire. Les priorités s'étalent de 0 à 255. Si plusieurs tâches sont de même priorité, elles sont exécutées à la queue leu leu. *Stack_size* est la taille de la pile de telle tâche (« mot » de 16 bits). Notez que quelques fonctions Microsoft (comme *printf()*) abusent d'un nombre démesuré de piles. S'il vous arrive de faire appel à des procédures de la bibliothèque Microsoft, il vous faudra des piles d'au moins 1 K (*stak_size* valant 512).

Les arguments suivants sont une liste de pointeurs, terminée par un *NULL,* qu'on passe à la tâche lorsqu'on la lance (cf. *infra*).

T_start() oblige à redistribuer. En l'espèce, si la tâche, que vous êtes en train de créer, est de priorité supérieure à celle de celle en exécution, elle doit alors prendre la main immédiatement.

Un pointeur sur le *TCB* venant d'être créé est habituellement renvoyé. Il servira ultérieurement à supprimer la tâche. Les valeurs d'erreur retournées sont *TE_TOOMANY* (le nombre maximal de tâches [32] est atteint) et *TE_NOMEM* (mémoire insuffisante pour créer la tâche).

Un modèle de création de tâche est donné dans l'exemple 2, page (). Ici, une tâche unique, dénommée *foo*(), est créée dans *main().* Son niveau de priorité vaut 10 et elle a une pile de 512 « mots » (1 K). Les arguments suivants sont passés à *foo()* au moment de son lancement. *Foo()* va afficher ses arguments et s'auto-détruire (elle affiche « Bonjour...»).

Notez qu'une tâche ne doit jamais s'achever par un retour normal : il est

```
tlmer( arg )
{
/* On suppose que les pointeurs et entiers sont ici de la même taille */


int i;

while( i )
{
for( i = 10000; —i >=  0 ; )
    ;

t_cli (\sc128\);
dv_putchar ( arg + '0');
t-sti(\sc128\);
}
}

maintask(\sc128\)
{
TCB *t1, *t2;

t1 = t_create ( timer,"1st timer", 100, 512, (void*)1, NULL);
t2 = t_create ( timer,"2nd timer", 100, 512, (void*)2, NULL);
t_perror("task1:", (int)t1 );
t_perror("task2:", (int)t2 );
t_wait( Queue2, 5 * t_second(\sc128\));
t_delete( t1 );
t_delete ( t2 );
t_delete ( NULL );
}
```

Exemple 8 : Le test 3, utilisé dans l'exemple 3

toujours préférable qu'elle s'auto-détruise. Si un retour est exécuté (explicitement ou, implicitement, parce qu'on est en bout de sous-procédure), *t_stop()* est alors immédiatement invoquée (avec un argument fantoche).

La priorité d'une tâche peut être modifiée à tout instant au moyen de l'appel à :

int t_chg_priority(tp,new_priority)
*TCB *tp;*
int new_priority;

Comme *t_create()*, cette procédure provoque une redistribution. Si une tâche attend un message, on la met aussitôt hors délai et on la réintègre dans la liste des instances. Une tâche peut modifier d'elle même sa priorité. Cette procédure renvoie normalement *TE_NOERR* mais, si la tâche n'existe pas, où si sa priorité est supérieure à 255, elle renvoie *TE_BADARG.*

Les tâches sont supprimées au moyen de :

int t_delete(task)
*TCB *task;*

qui libère du même coup toute la mémoire associée à *task.* Notez que la mémoire, que la tâche s'est d'elle-même attribuée (via un appel *malloc()* ou équivalent), n'est pas libérée - seule l'est la mémoire que *t_create()* attribue (pour la pile et le reste). Une tâche s'auto-détruit par un appel à *t_delete(NULL). T_delete()* provoque une redistribution lorsque c'est la tâche en cours qui est supprimée. Les valeurs de retour sont *TE_NOERR,* quand ça c'est bien passé, et *TE_BADARG,* si la tâche n'existe pas.

Les messages

Trois procédures servent à créer des files de messages (« boîtes postales ») et à passer des communications entre

tâches. Les files sont créées au moyen de :

```
T_QUEUE *t_makequeue(size)
int size;
```

où *size* est le nombre maximal de messages pouvant être en file d'attente. Cependant, il peut y avoir tout nombre de tâches en attente. Normalement, un pointeur sur la file d'attente est renvoyé, mais TE_NOMEM l'est si l'on ne dispose pas suffisamment de mémoire. Ces files sont reliées au sein d'une liste linéaire, où l'on recherche les tâches hors délai à chaque top d'horloge; aussi vaut-il mieux créer en premier les files les plus actives.

Les messages sont envoyés d'une vers une file d'attente au moyen de :

```
int t send(q, msg)
T_QUEUE*,q;
void * msg;
```

où *q* est un pointeur précédemment renvoyé par un appel à *t_makequeue()*, et où *msg* pointe sur le message. Le pointeur, et non le message, est emmagasiné ; ainsi les messages peuvent-ils être *ad libitum*. Typiquement, *msg* peut aussi bien pointer sur une structure où sur une chaîne. Le message, lui, doit résider en mémoire statique.

S'il n'y a pas de tâche en attente dans la file, *t_send()* fait aussitôt retour ; sinon, le message est relié à la tâche et l'ordonnateur (2) est sollicité; cela signifie que, si la tâche en attente est de plus haute priorité, cette dernière prendra le pas sur celle envoyant le message. *TE_NOERR* est normalement renvoyé. Les erreurs retournées sont *TE_BADARG*, si l'argument *q* est défectueux, et *TE_QFULL*, si la file est saturée.

L'autre aspect de la messagerie est :

```
void *t_wait(q,timeout)
T_QUEUE *q;
int timeout;
```

dont une tâche se sert pour attendre dans la file l'arrivée d'un message. *Q* est un pointeur précédemment retourné par l'appel à *t_makequeue()* et *timeout* est la valeur du délai. La tâche reste en attente durant *timeout* tops de l'horloge système avant que d'être réintégrée dans la liste d'instances (pour mémoire, l'horloge ne délivre pas de tops tant que l'ordonnateur est bloqué). Le délai maximal est de 32767 tops d'horloge (soit environ une demie heure à la fréquence de 18,2 tops par seconde). Si le délai vaut 0, *t_wait()* fait immédiatement retour (sans redistribution) s'il n'y a pas de message en instance dans la file d'attente.

Les demandes de messages sont mises en file d'attente dans l'ordre où elles sont reçues, sans considération de priorité. J'ai fait cela pour deux raisons : parce que c'est commode et parce que, dans bien des applications, les tâches de priorités différentes n'attendent pas dans une même file.

Si un message se trouve dans la file d'attente, *t_wait()* renvoie aussitôt le pointeur du message (le même que celui passé dans *t_send())* sans faire de redistribution; sinon, la tâche active est suspendue (ôtée de la liste des instances) jusqu'à l'arrivée du message.

Normalement, un pointeur de message est renvoyé. Les valeurs d'erreur renvoyées sont *TE_TIMEOUT* (si le délai est passé ou si l'argument timeout vaut 0 et qu'il n'y a pas de message en instance) et *TE_NOTASKS* (si la tâche en cours est la seule à fonctionner - l'impasse à coup sûr).

```
double mean(reset, data)
int reset;
double data;
{
  static double xhat, ki;

 return reset ? (ki = xhat = 0)
      : (xhat += (data \sc0\ xhat) / ++ki)
      ;
}
```

Exemple 9 : Une fonction C intégrant l'algorithme de la figure 1

Bloquer et libérer

Quatre sous-procédures sont fournies pour pourvoir au blocage : void t_cli();, *void t_sti();, void t_block(); et void t_release();*.

T_cli() et t_sti() désactivent et réactivent les interruptions, respectivement. A utiliser avec précaution. Les interruptions ne doivent jamais être désactivées longtemps. Elles ne doivent jamais l'être quand on se sert des fonctions d'E/S du DOS.

T_block() et *t_release()* sont plus sûres. *T_block()* inactive l'ordonnateur mais non l'interruption normale d'horloge. En fait, la tâche en cours garde le contrôle du système une fois que *t_block()* a été appelée. Parce que les interruptions n'ont pas été inactivées, c'est une procédure bien plus sûre à l'usage que *t_cli()*. Il y a cependant un piège. Si l'interruption normale survient, alors que les interruptions normales ont été inactivées, la machine exécutera les interruptions aussitôt celles-ci réactivées. Ce n'est pas le cas avec *t_block()*. En fait, l'ordonnateur n'a pas à savoir si une interruption est survenue tandis qu'il était bloqué. En conséquence, une courte boucle telle que :

```
while( *str )
{
        t_block();
        putchar( *str++ );
        t_restore();
}
```

ne fonctionnera pas de la manière attendue. Parce que très peu de temps s'écoule entre les appels à *t_restore()* et à *t_block*, il y a des chances pour qu'aucune interruption ne se manifeste tandis que l'ordonnateur est actif. En fait, l'exemple qui suit fonctionnera probablement d'une même manière que le précédent :

```
t_block();
while( *str )
putchar( *str++ );
t_restore();
```

Notez qu'un *t_cli()* peut déconnecter l'horloge système du DOS (le temps sera perdu) tandis que les interruptions sont inactives; *t_block()* n'a pas cette restriction.

La dernière sous-procédure de gestion est *void t_yield();*. Elle met en sommeil la tâche en cours et ranime la tâche de plus haute priorité suivante. Notez que

la tâche en cours est toujours suspendue, même s'il s'agit de celle de la plus haute priorité. Elle reprendra la main, dans ce cas, dès la prochaine interruption d'horloge. *T_yield()* est d'abord utilisée lorsque l'ordonnateur est bloqué ou quand on exécute un système fondé sur la « préséance ». Elle est aussi utile quand une tâche de haute priorité n'a pas grand'chose à faire et n'a pas besoin d'occuper le *CPU*. *T_yield()* retourne *TE_NOERR,* quand la libération s'est bien passée, et *TE_NOTASKS,* s'il n'y a aucune autre tâche à qui passer la main.

Notez que *t_yield()* ne retourne pas tant qu'elle n'a pas repris contrôle du système. Ainsi le flux normal de contrôle se déroule-t-il comme suit:

1. Une tâche appelle *t_yield().*
2. La tâche suivante est activée.
3. Puis elle est suspendue.
4. Et le contrôle revient à la tâche de départ et *t_yield()* renvoie *TE_NOERR.*

Statistiques

Diverses procédures sont fournies, à fin de statistiques et d'« épuçage ». Deux servent à épucer:

```
t_tprint( t )
TCB *t;

t_qprint( q )
T_QUEUE *q;
```

T_tprint() reçoit un pointeur de *TCB* et produit intelligiblement le *TCB* sur le périphérique standard; *t_qprint()* fait de même avec les files.

Quelques procédures statistiques sont utiles aussi: *long t_numint();* renvoie le nombre d'interruptions non bloquées et long *t_numblk();,* le nombre de celles bloquées. Enfin, *t_sstats()* produit diverses statistiques relatives à l'ordonnateur (le nombre de fois qu'il a été sollicité, le nombre de dépassements de délai et le nombre d'échanges de contextes). Cette procédure ne peut être appelée depuis une tâche (parce qu'elle se sert de *printf()).*

Divers

int t_second()
Renvoie le nombre de tops par seconde du système en fonction du facteur d'accélération passé à *t_start(). Elle renvoie 0 si le facteur d'accélération valait 0. Dans ce cas, un appel à t_wait()* n'est jamais en dépassement de temps.

*char *t_errlist[]*
Fonctionne comme *l'errlist[]* normale que gèrent bien des compilateurs C. Indicée par code d'erreur, elle dépose ses évaluations dans une chaîne contenant un message d'erreur approprié.

t_iserr(x)
Est vraie si *x* est un code d'erreur; sinon elle est fausse.

t_perror(str,errcode)
*char *str;*
Affiche un message d'erreur. Si *errcode* est un code d'erreur, elle affiche le message approprié; sinon, elle affiche *status XXX,* où *XXX* est la représentation décimale de *errcode.*

t_printf(fmt, ...)
*char *fmt;*
Fonctionne comme *printf(),* si ce n'est qu'elle met au gabarit avant d'afficher quoi que ce soit.

T_PRIORITY(a,b)
*TCB *a,*b;*Cette *macro,* qui se trouve dans kernel.h, compare les priorités des deuxtâches. Elle renvoie une valeur négative si la priorité de *a* est inférieure à celle de *b*, 0 si elles sont égales, et une valeur positive si la tâche *b* est d'une priorité inférieure à celle de la tâche *a*. Notez que la priorité d'une tâche est établie en fonction et de la priorité passée à *t_create()* et du moment où la tâche a été suspendue pour la dernière fois. En d'autres termes, si deux priorités sont expressément égales, la tâche la plus récemment suspendue est considérée moins prioritaire que l'autre. Ce mécanisme rend l'ordonnancement à la queue leuleu facile à intégrer, les tâches étant mises en bout de liste au fur et à mesure de leur suspension.

Quelques exemples

Considérons un petit nombre d'exemples. En premier lieu, *main()* va créer une couple de files et lancer le multitâches. Je me servirai d'un module *main()* commun pour tous les exemples, lesquels figurent sous Exemple 3. Un test parmi trois est effectué, selon la valeur de la macro *TEST*. Deux files d'attente, toutes deux de longueur 2, sont créées en début de sous-procédure. Puis quelques tâches sont créées et l'exécution multi-tâches est lancée au moyen d'un appel à *t_start().* Enfin, après que *t_start()* aura rendu la main, diverses statistiques seront affichées.

Considérons maintenant les tests individuellement. Le test 1 expérimente le mode « sans préséance » et met en évidence la passation de messages. Les deux tâches, *sam()* et *dave(),* figurent sous Exemple 4 et Exemple 5, ci-contre(). Elles affichent toutes deux leurs arguments de démarrage (provenant de l'appel de début à *t_start())* puis reçoivent et émettent quelques messages. Notez que, bien que j'aie passé des chaînes en tant que messages, on peut passer des pointeurs dans tous les cas. Les appels à *t_wait()* et *t_yield()* entraînent chacun le transfert du commandement d'une tâche à l'autre. De même, *sam()* tente intentionnellement de saturer de messages la file d'attente afin d'observer le bon fonctionnement du mécanisme d'erreurs. Ce que le programme sort figure dans l'Exemple 6, ci-contre(). Notez que toutes les valeurs statistiques, figurant au bas, sont à 0 en raison du mode « sans préséance » du système.

Le texte suivant est un petit compte-temps. Il se sert de la tâche *idle(),* que j'ai analysée précédemment. La tâche *timer()* figure en Exemple 7. Elle affiche un message de départ et entre dans une boucle for qu'elle exécute cinq fois. *L'appel* à *t_wait()* est en dépassement de temps au bout d'une seconde, personne n'envoyant de message dans le file d'attente. C'est pourquoi le compte-temps imprimera cinq points d'exclamation, un à chaque seconde, puis s'auto-détruira.

Le troisième test figure en Exemple 8, ci-après(). Il met en évidence plusieurs choses. En tout premier lieu, *main()* crée une tâche seulement, *maintask(),* qui, à son tour, crée deux tâches supplémentaires. Notez, cependant, que ces deux tâches utilisent le même code ! C'est possible parce que chaque tâche possède sa propre pile. Plusieurs tâches peuvent avoir en commun le même code pourvu qu'elles n'aient pas recours à des variables statiques (les variables locales se trouveront sur des piles physiquement distinctes). Ici, la tâche s'identifie elle-même, en examinant son argument, puis l'imprime à chaque fois. La sous-procédure *dv_putchar()* est une fonction de sortie vidéo directe, toutes ces fonctions « font avec ».

...Suite p. 75

Les bibliothèques de liaisons dynamiques DLL sous Windows

Partager le code via les DLL est un moyen efficace de pallier la surcharge dans un environnement multi-tâches.

par **Margaret Johnson** et **Mark Solinski**
traduit par **Philippe Dalléas**

Beaucoup de gens étant sensibles aux avantages d'OS/2, les Dynamic Link Libraries (bibliothèques de liaisons dynamiques) - connues aussi sous les vocables dynalinks ou DLL - sont devenues un sujet d'actualité. Peu de personnes, toutefois, se sont rendu compte que les DLL sont depuis plus de quatre ans à la disposition des programmeurs en tant qu'élément de l'environnement exécutif de Windows Microsoft. En fait, la plus grande partie de l'environnement de Windows est intégrée en tant que DLL. Cet article analyse les propriétés des DLL et fournit aussi un exemple clair d'une DLL qu'ont peut intégrer à toute application Windows. Notre voeu est que les développeurs, créant actuellement des librairies prêtes à l'emploi pour MS-DOS et projetant de nouveaux produits pour OS/2, prennent conscience que les DLL de Microsoft pour Windows constituent l'une des plates-formes de développement de leurs produits.

Les DLL comparées aux bibliothèques prêtes à l'emploi

Pour produire un fichier exécutable, dans le monde MS-DOS, le programme est compilé et relié à des bibliothèques externes ou modules « objets ». La plupart du temps, les procédures contenues dans les bibliothèques sont « génériques », consistant en fonctions souvent utilisées - par exemple, la bibliothèque prête à l'emploi livrée avec le compilateur. Chaque programme appelant une fonction d'une bibliothèque contient en soi une copie de cette fonction. Dans le cadre typique du système monotâche DOS, et sans tenir compte de la taille des fichiers exécutables, les bibliothèques prêtes à l'emploi sont parfaites.

Mais dans un système multi-tâches tel Windows, il n'est pas efficace que chaque application porte en soi une copie d'un code commun. Windows est un environnement exécutif installé par dessus DOS, créant pour ses applications un environnement multi-tâches sans « préséance »et géré sur événements. Ces « bagages » de code commun peuvent provoquer de sérieuses difficultés en ce que Windows, DOS et vos applications doivent se partager les fameux 640 Ko de la mémoire (si votre système n'ait pas de mémoire étendue et que le pilote himem.sys n'ait pas été installé).

Windows soulage de ces congestions en permettant à plusieurs instances de la même application de partager le même morceau du code, mais, malheureusement, s'il s'agit d'applications différentes, le code n'est pas partagé. Par exemple, si chaque application appelle la même fonction d'une bibliothèque pour lisser des données, Windows attribue à la fonction de lissage un espace de code pour chaque

[SERVICE LECTEURS : INTERESSANT : 13 / MOYEN : 14 / MAUVAIS : 15]

```
Un exemple de cela pourrait être une DLL contenant les procédures
établissant le graphe des valeurs des variables. Ordinairement, pour exploiter
une telle fonction, on procède aux appels d'initialisation,
de chargement des variables, de tracement du graphe puis de vidage.
Déposer les variables du graphe dans une mémoire « globale »
prend en compte le fait que chaque application faisant
cette série d'appels possède  sa propre zone de données
où conserver ses variables.
Si deux applications (A et B) coexistent, que l'application A
fait les appels pour charger les variables du graphe et
que B fasse la même chose  avant que A n'ait pu faire l'appel
du tracement des variables,  celles dont se sert la DLL
contiendront les options de B et non celles de A.
```

Exemple 1 : **Exemple d'une ressource, créée par l'utilisateur, appelée CAVEATS. ASC.**

application. Comme chacun qui a fait fonctionner Windows en multi-applications le sait, la mémoire est à Windows ce que l'eau est au désert : une ressource précieuse à ne pas gaspiller.

Si une fonction se trouve dans une DLL, il n'y en a alors qu'une seule copie de chargée, quelque soit le nombre d'applications y faisant appel. Ceci raccourcit les applications exécutables du fait que le code commun n'y est pas incorporé. De même, le code est dynamiquement chargé au moment de l'exécution, donc, tout le temps que les paramètres passés lors d'un appel de fonction resteront inchangés, vous pourrez à tout moment modifier le contenu d'une fonction d'une DLL sans avoir à recompiler l'application ni à en reconstruire les liaisons. En conséquence, l'utilisation judicieuse des DLL vous permet de personnaliser et mettre à jour vos applications sans devoir produire de versions entièrement nouvelles.

Charger la bibliothèque au moment de l'exécution offre d'autres occasions intéressantes. Quelques progiciels - par exemple, Actor de WhiteWater, Excel de Microsoft ou SQL Windows de Gupta - peuvent comporter des fonctions DLL dans leurs programmes. Cette faculté permet aux développeurs de se servir de leurs propres fonctions les plus intimes dans des applications existantes extrêmement puissantes. Mais ce n'est pas tout! Il n'est pas toujours nécessaire que les DLL contiennent du code ou des données. Elles peuvent être aussi constituées de ressources du système, telles que des fontes, des images point par point, des icônes et des curseurs — un grand moyen pour personnaliser des applications sans avoir à distribuer le code « source », un « plus »pour les développeurs de bibliothèques.

Problèmes dans la création de DLL

Comme dans la vie, on n'obtient pas de profit sans peine et c'est assurément vrai quant aux DLL. Il y a deux grandes différences entre le code d'une DLL et celui d'une bibliothèque prête à l'emploi classique pour MS-DOS. La première est sur le bout de la langue de quiconque s'est risqué dans le domaine des DLL - c'est la fameuse expression SS!=DS. La seconde a trait à l'utilisation de variables « globales ».

Normalement, quand le code C est compilé, on prend en charge le fait que le segment des données (DS) soit le même que celui de la pile (SS). C'est valable quand on passe à une fonction l'adresse d'un paramètre, dans le cadre du « petit modèle » (un segment de code et un segment de données) et du « moyen » (plusieurs segments de code et un de données), parce qu'on ne passe que les 16 bits de décalages d'adresses.

Les DLL ont toutefois leurs propres segments de données et se servent de la pile de l'appelant. Dans ce cas, le segment de la pile doit différer du segment des données (SS!=DS) et les décalages de 16 bits ne constituent pas des adresses valables parce que les variables « automatiques »- paramètres et variables « locales » - sont empilées, tandis que toutes les autres - variables « globales » et « statiques »- sont contenues dans le segment de données. Cela signifie qu'une adresse passée à une fonction doit contenir et le segment et le décalage, sous forme de pointeurs far et, qu'en conséquence, bien des fonctions des bibliothèques prêtes à l'emploi au standard C sont inutilisables. Ces fonctions sont inventoriées en annexe du kit de développement pour Windows SDK de Microsoft. En général, toute fonction d'E/S tamponnée (fprintf, fscanf et sscanf, etc.), toute fonction « exec » (comme execlp, execv) et toute fonction « spawn » (par exemple, spawnl, spawnv) se trouvent hors domaine.

Le second problème vient de la mentalité et de l'habitude que l'on a acquis en écrivant les logiciels mono-tâches. Parce que de tels programmes n'ont pas besoin d'être « réentrants », il est courant d'émailler généreusement le code de variables « globales ». Cette façon de faire peut être une source d'ennuis pour la personne souhaitant convertir le code existant pour qu'il fonctionne dans une DLL. Bien que dans Windows le code ne soit pas « réentrant », les données le sont, par ce que les DLL n'ont qu'un seul segment de données, lequel est partagé par toutes les applications. Lorsqu'on retrouve une variable dans telle fonction, on ne doit pas s'attendre à ce qu'elle ait la valeur que lui a donnée une autre fonction de la DLL.

Un exemple de cela pourrait être une DLL contenant les procédures établissant le graphe des valeurs des variables. Ordinairement, pour exploiter une telle fonction, on procède aux appels d'initialisation, de chargement des variables, de tracement du graphe puis de vidage. Déposer les variables du graphe dans une mémoire « globale » prend en compte le fait que chaque application faisant cette série d'appels possède sa propre zone de données où conserver ses variables. Si deux applications (A et B) coexistent, que l'application A fait les appels pour charger les variables du graphe et que B fasse la même chose avant que A n'ait pu faire l'appel du tracement des variables, celles dont se sert la DLL contiendront les options de B et non celles de A.

La plupart de ces problèmes sont inhérents aux systèmes multi-tâches et à ceux à « code partagé » et ne sont pas seulement des « gags » propres à Windows. Les solutions à ces problèmes, incorporées grâce à une conception soignée du logiciel, seront certainement profitables aux programmeurs désirant porter leurs applications et bibliothèques dans les environnements multi-tâches, OS/2 inclus. L'exemple qui suit montre comment un utilitaire d'assistance simple, qu'on peut enrichir, peut être associé à toute application et personnalisé sans avoir à recompiler l'application principale ni à en reconstruire les liaisons.

La bibliothèque d'assistance

La bibliothèque d'assistance que nous avons créée suit la convention mise en exergue dans le « Microsoft Windows Application Style Guide ». Toutes les applications suivant ces règles auront le même aspect et présenteront une solide interface d'utilisation (clavier, souris, écran et autres ports). Des phrases comme « Toutes (vos) applications ... »,«... même aspect ...» et

«... solide interface ...» vous rappellent sans doute certains sons de cloche. Ce sont là quelques uns des critères usuels de production de bibliothèques de fonctions. Les bibliothèques « dynamiques » constituent même un meilleur choix parce qu'il arrive que les habitudes changent, que les nouveaux standards soient incorporés dans une nouvelle bibliothèque et qu'ils en deviennent immédiatement disponibles pour toutes les applications qui se servaient de l'ancienne bibliothèque, sans qu'on ait à recompiler ces dernières ni à en reconstruire les liaisons.

Cette discussion met en évidence une autre propriété des DLL — la faculté d'isoler le code de personnalisation dans différentes bibliothèques. Ainsi, les développeurs n'ont-ils à publier que les spécifications des fonctions relatives aux procédures contenues dans la bibliothèque. Microsoft use de cette technique pour ses pilotes de matériels. Les fabricants d'imprimantes, quant à eux, peuvent écrire des DLL, tirant profit des meilleures propriétés de leurs machines et fournir à Windows le même « aspect fonctionnel ». Le fichier helplib.c (listing Un) définit le protocole « public » de la bibliothèque d'assistance. Les deux fonctions, que nous avons conçues, sont Topics() et Screen(). Topics affiche la liste des sujets d'aide disponibles; Screen affiche le texte relatif à un sujet particulier. Il y a une relation biunivoque entre un affichage et un sujet. Cette relation est établie par l'usage dans le fichier ressource helplib.rc (listing Deux) du même intitulé pour les textes et les ressources définies par l'utilisateur. Par exemple, l'intitulé du texte relatif aux « avertissements » est le même que celui de la ressource définie par l'utilisateur, à savoir CAVEATS. Le texte à afficher se trouve dans le fichier ASCII CAVEATS.ASC (cf. Exemple 1). Ce fichier peut être créé par tout éditeur de textes. Il y a un fichier par affichage. Chaque application partageant la bibliothèque d'assistance aura sa liste séquentielle des intitulés (et textes associés) relatifs aux sujets de l'assistance. Les exemples 2, 3 et 4 répertorient d'autres ressources définies par l'utilisateur, dont se sert notre bibliothèque.

En entrée, les deux fonctions reçoivent une variable de type LPSCREEN (comme défini dans helplib.h, listing Trois). Ce type correspond à un pointeur far vers un type SCREEN défini en tant que structure contenant les variables wStart, wEnd, et wScreen. Topics() se sert des valeurs de wStart et wEnd pour délimiter la zone de chaînes de caractères à charger. Topics() et Screen() se servent toutes les deux de wScreen pour référencer les ressources que sont les fichiers ASCII définis par l'utilisateur. Toutes les fonctions « publiques » (celles qui, dans une bibliothèque, peuvent être appelées de l'extérieur) doivent être déclarées en tant que FAR PASCAL. Déclarer une fonction comme FAR lui permet d'être accessible à une autre application n'ayant pas le même segment de code. Le mot PASCAL précise qu'on utilisera la forme d'appel du Pascal, ce qui signifie que les paramètres seront déposés sur la pile selon l'ordre de gauche à droite, au lieu de droite à gauche comme en langage C.

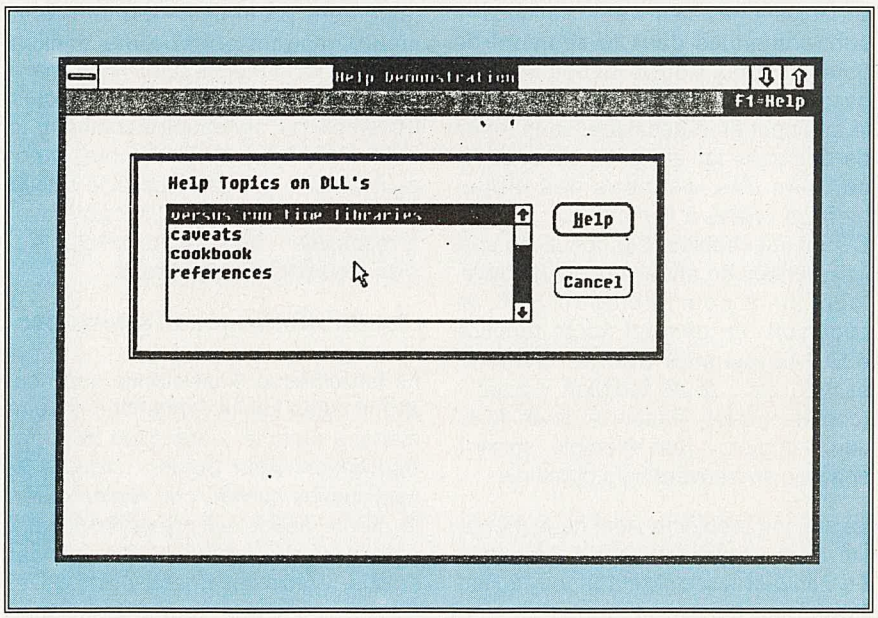


Figure 1 : **La boite de dialogue qui apparaît lorsque Topics () est appelée.**

La figure 1 montre la boîte de dialogue apparaissant quand Topics() est appelée. Elle affiche une boîte de listage contenant les chaînes précitées et les deux boutons-poussoirs « Help » et « Cancel ». La fonction Screen() est appelée quand on veut obtenir l'assistance propre au sujet illuminé. Comme le montre la figure 2, Screen() affiche une boîte de dialogue où apparaît le texte d'assistance dans une fenêtre à défilement surmontant les quatre boutons-poussoirs « Topics », « Next », « Previous » et « Cancel ». Pousser « Topics » renvoie la boîte de dialogue « Topics ». Pousser « Next » affiche le texte du sujet suivant dans la liste de la boîte de dialogue « Topics »; et pousser « Previous », celui du précédent. On peut aussi appeler directement Screen() pour afficher un texte d'assistance particulier. Le protocole « privé » de la DLL est installé dans le fichier helpdlg.c (listing Quatre). Les fonctions ScreenDlgProc, ScreenWndProc et TopicDlgProc préparent les messages que Microsoft Windows envoie aux boîtes de dialogue. En l'espèce, nous sommes beaucoup plus concernés par la messagerie de l'interface d'utilisation, comme dérouler le contenu d'une fenêtre ou la déplacer, et celle du système, comme réafficher une partie de l'écran qui est maintenant visible parce qu'elle a été ramenée par devant une fenêtre la masquant jusque là. Une complète analyse du système de messagerie de Microsoft Windows sort du champ de cet article; nous recommandons aux gens qui s'intéressent au sujet de se procurer un exemplaire de Programming Windows, par Charles Petzold (Microsoft Press, 1988) — une aide précieuse pour tous les progmeurs Windows.

Un autre point important du protocole « privé » de la DLL est la maintenance des informations d'affichage. La « réentrance » est une « caractéristique » des DLL. Cela complique le sujet si l'on a besoin de maintenir des données « globales ». Un moyen de protéger les données « globales » est d'attribuer des octets supplémentaires à la structure « window » au moment de déclarer la classe « window ». Nous avons obtenu cela en réglant la zone cbWndExtra sur sizeof(HANDLE) octets quand on déclare la classe « window » de l'écran d'aide dans libinitc.c (listing Cinq). Quand ScreenWndProc() reçoit un message WM_CREATE, elle attribue un « placard » de mémoire « locale », où conserver les informations d'affichage de la fenêtre, et en dépose la clef dans

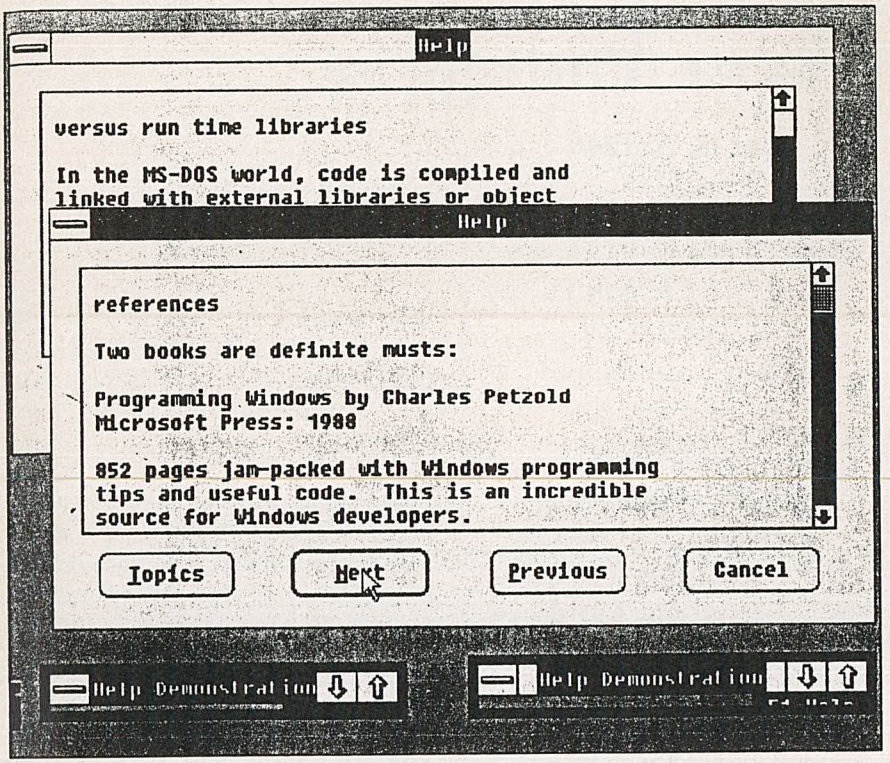


Figure 2 : La fonction Screen () fait afficher la boite de dialogue qui présente le texte dans une fenêtre, en permettant des manipulations avec des boutons.

la structure « window » à l'emplacement 0. Cela permet une recherche commode, lorsqu'arrive le moment où l'on a besoin des données d'affichage de la fenêtre d'aide.

Les DLL diffèrent des applications en ce qu'elles ne peuvent ni ne doivent invoquer l'initialisation du « C run-time Windows ». Toute initialisation, dont elles auraient besoin, doit leur être fournie par le développeur au moyen, ordinairement, d'un cours programme en Assembleur, appelant une fonction C dans la bibliothèque, pour y procéder. libintc.cc et libinita.asm (listing Six) assurent ce service.

Le dernier élément du puzzle constituant notre bibliothèque d'assistance est le fichier de définition des modules. Son premier objectif est de définir lescaractéristiques des données et les segments de code, les piles locales et les fonctions exportées. S'il y avait aussi un fichier de définition de module pour une application Windows, il inclurait alors un paramètre de taille de pile, ce qui n'est pas utile dans une DLL, parce qu'elle se sert de la pile de l'application appelant. C'est aussi parce que les bibliothèques n'ont qu'un seul segment de données, que la ligne de déclaration DATA doit contenir l'option SINGLE.

La section EXPORT définit les fonctions accessibles de l'extérieur. Bien sûr, Screen() et Topics() figurent dans cette liste, mais on y trouve aussi les fonctions ScreenDglProc, Screen WndProc et TopicsDlgProc. C'est indispensable parce que ces procédures de la DLL sont appelées par Windows pour traiter ce qu'entre l'utilisateur, et c'est la raison pour laquelle ces fonctions sont déclarées en tant que FAR PASCAL. L'éditeur de liens LINK4 utilise le fichier de définition de module pour ajouter les informations indispensables au fichier .EXE, lequel, en fonctionnement, permet d'établir les relations avec les procédures de la bibliothèque et de les exécuter.

Il n'est pas toujours nécessaire que les DLL contiennent du code ou des données. Elles peuvent n'être composées que de ressources telles les fontes, les images point par point, les icônes ou les curseurs.

Le fichier constructeur helplib (listing Huit) se sert des fichiers décrits précédemment pour créer les fichiers helplib.lib et helplib.exe de la bibliothèque. Les commutateurs de compilation nécessaires pour une DLL sont -Alnw (ou -Asnw, s'il s'agit du « petit » modèle). Ils sont indispensables pour que SS!=DS.

Tester la bibliothèque

La bibliothèque d'assistance, que nous fournissons, donne un résumé de cet article et une description pas à pas de la façon de créer une bibliothèque. Nous avons aussi inclus deux exemples montrant comment utiliser la bibliothèque. Le premier, helpdemo . exe, et son code source (listings Neuf à Dix-huit) montre à quel point il est facile d'appeler une DLL depuis une application Windows. La fonction HelpDemoMsg() définit les informations d'affichage et appelle la fonction Topics() dans la DLL chaque fois que la touche F1 est pressée ou que l'élément « Help! » du menu est activé.

Le second exemple, helplib.act (listing Dix-neuf) est écrit en Actor, langage incrémental compilé destiné au développement d'applications Windows. Cet exemple montre comment on peut, à l'exécution, spécifier différentes DLL. Tant qu'elle connaît les fonctions contenues dans une bibliothèque, une application peut la charger, en effectuer les fonctions puis la relâcher. Actor offre le moyen de changer de bibliothèque « en plein vol » et de voir comment de tels changements influent sur le comportement de vos applications.

Conclusion

Les bibliothèques de liaisons dynamiques constituent un puissant outil de réutilisation de code. Grâce à une organisation modulaire correcte, les applications peuvent être personnalisées sans les recompiler ni en reconstruire les liaisons. Les DLL fonctionnent mieux quand elles sont appelées pour traiter les données et repartir.

C'est parce que les DLL sont partagées par toutes les applications du système qui les appellent, bien que les données « globales » ne puissent être maintenues. Une façon de pallier cette limitation est d'emmagasiner les informations propres à une fenêtre dans sa structure de données.

Nous espérons que cet article vous aura donné une idée de la souplesse des DLL. Les développeurs ayant acquis des DLL pour Microsoft Windows auront eu un avant-goût de la programmation OS/2. Ceux qui développent des bibliothèques tierces pour les vendre au monde des programmeurs PC, trouveront une grande audience auprès de la « communauté » Microsoft Windows et auront aussi une plate-forme commode pour porter leurs bibliothèques sous OS/2.

Les bibliothèques de liaisons dynamiques DLL sous Windows : Copyright **Dr. Dobb's Journal** M&T Publishing Inc. Redwood City, CA USA 1990. All right reserved.

Un Système Multiposte Idéal Pour Une Entreprise

par *G. Korwin*

LA SCENE

Les systèmes multiutilisateur s'adressent essentiellement aux entreprises (privés ou publics). C'est l'entreprise, lieu d'activité des individus en commun, qui est son principal client. Les systèmes basés sur les équipements micro-informatiques s'adressent plus précisément aux PME/PMI: des structures qui emploient de zéro à 100 salariés. Des entreprises plus grandes sont généralement déjà équipés en systèmes informatiques « lourd », basés sur mini ou mainframes.

LE LEVE DE RIDEAU

Dessinons rapidement une brève image des PMI Françaises. Selon un récent rapport de SOFRES, il y a en France actuellement 2.166.900 entreprises, correspondant au profil d'une PMI. La démographie de ces entreprises montre leur jeunesse et le dynamisme des créateurs : 700.000 d'entre elles ont été créées après 1980, leurs dirigeants ont en moyenne 40 ans. Sur ces 700.000 entreprises les plus jeunes, 80% comptent moins de 3 salariés, 125.000 moins de 50 et 2900 emploient plus de 50 personnes. Elles ont réalisé en 1990 1.200 millions de francs de chiffre d'affaires (14% de l'ensemble de PME). Ce créneau de « jeune PME » est aussi en pleine mutation: pendant la même période (1980/1990) 2 millions de PME ont été créées mais 1.8 millions ont disparu... Bien que 88% déclarent être déjà « informatisées », il y a encore beaucoup à faire.

NOTRE ACTEUR VEDETTE

Pour illustrer notre comparaison, nous avons imaginé une société fictive qui entreprend d'informatiser ses services commerciaux et administratifs. Notre société imaginaire s'appelle Courtebotte Industrie et Distribution SA (CID SA). Elle a été créée tout juste il y a un an, c'est une entreprise dont l'activité se situe à mi-chemin entre le négoce et la fabrication. Située à St Jean de Blagnac, en pleine campagne bordelaise, elle emploie actuellement 3 salariés : le créateur et son gérant, Monsieur Pierre Courtebotte, sa fidèle secrétaire et comptable Aniece et un commercial, André.

Le but de cet article est de faire le point de l'état d'art dans le domaine des équipements multipostes micro-informatiques. Vous allez dire: encore une « moulinette à ASCII ». Non. Ce qui nous intéresse, c' est l'accès à plusieurs aux ressources disponibles, et non un logiciel d'application précis. Nous n'allons pas parler de programme de la compta ou gestion de fichiers mailing mais de choix de topologie, moyens de transmission, équipements nécessaires et savoir faire. Le dossier présenté ci-dessous est envoyé simultanément aux principaux fournisseurs de solution multipostes en France. Nous allons publier leurs réponses.

L'entreprise s'occupe de vente des vins de qualité, aux clients particuliers et points de vente spécialisés en France et à l'étranger. A cause de sa spécialisation, qualité de produits et dynamisme de son équipe, les affaires explosent. Le chiffre d'affaires double tous les trois mois : cent mille francs, deux mille francs, actuellement huit cent mille francs. Le premier exercice doit se fermer avec un chiffre d'affaires qui va dépasser trois millions de francs.

Évidemment, l'activité commerciale la mieux développée peut tomber à l'eau si l'organisation du travail administratif ne suit pas. Le désordre est le pire ennemi de chaque bon gestionnaire. Notre ami Jean Courtebotte le sait bien et c'est pour cette raison qu'il a entrepris, depuis le début, l'informatisation à l'outrance. Il n'y a pas de machine à écrire chez lui. A part une chaise, un bureau et un téléphone, chaque personne dispose aussi d'un ordinateur et des outils nécessaires comme le traitement de texte, PAO ou programme de comptabilité. Jean, habitué depuis le début à un PC, travaille avec un AT 386, il utilise le traitement de texte, un tableur et une base de données. La secrétaire comptable dispose d'un AT 286, pour suivre les commandes des clients et tenir la comptabilité. Le vendeur, qui a un don certain de communication, utilise un Mac - l'environnement graphique Apple correspond parfaitement à son esprit créatif et les outils PAO lui permettent de créer des documents publicitaires originaux.

LA MONTEE EN PUISSANCE

Tout serait au mieux dans le meilleur des mondes si... Si les affaires ne se développaient pas à un taux record. Les campagnes de promotion, publicité et mailings directs ont déclenché une véritable avalanche de commandes. Les vins de CID SA se vendent bien : par correspondance et par téléphone, l'équipe reçoit quotidiennement une vingtaine de commandes. Il faut les saisir, livrer, comptabiliser, émettre des bons de livraison et factures, enregistrer les paiements, relancer les comptes débiteurs etc. Chaque commande, dont le montant moyen se situe aux alentours de mille francs, génère au moins 4 opérations comp-

(Suite page 73)

[SERVICE LECTEURS : INTERESSANT : 37 / MOYEN : 38 / MAUVAIS : 39]

ENVIRONNEMENT MULTI :

REDECOUVERTE DE L'ASSEMBLEUR

par **Philippe Dalléas**

Notre entreprise se consacre à la recherche et au développement de logiciels de gestion. Dans ce contexte limitatif et dès la fin de l'année 1983, nos travaux nous ont notamment conduits à étudier divers systèmes d'exploitation et langages de développement répondant à des critères tels multi-tâches, multi-postes, portabilité, compatibilité des données et des applicatifs, capacité d'adressage dans les fichiers de données à accès direct, compétitivité économique, productivité, fiabilité et performances: en bref il nous fallait des moutons à cinq pattes !

Depuis 1983, un an et demi de travail et 110.000 Frs d'investissements en logiciels systèmes et langages nous ont amenés à jeter notre dévolu sur Concurrent CP/M puis Concurrent DOS (actuellement Multiuser DOS), système d'exploitation produit par DIGITAL RESEARCH offrant l'originalité d'exploiter indifféremment et simultanément des programmes provenant des opulentes bibliothèques DOS et CP/M-86 ou MP/M-86 d'une part; et nous permettant de nous mettre au diapason de nos grands confrères européens tels l'Angleterre et l'Allemagne où une machine multi sur trois, environ, est gérée par Concurrent DOS, d'autre part. Pour clore quant aux systèmes d'exploitation, nous avons particulièrement apprécié le réseau local NOVELL remarquable par ses performances globales et sa fiabilité.

A l'inverse, nous avons été dégoûtés par l'environnement Xenix (beaucoup plus grossier que gaulois!) du fait de sa médiocrité de performances, de la rareté, voire de l'inefficacité de ses utilitaires, et de sa documentation absconse.

Pour ce qui est du langage de développement, notre crible nous a extrait une perle, DATAFLEX, qu'un de nos amis avait découvert aux Etats-Unis et nous avait rapporté.

Mais même une perle peut avoir des défauts. En l'occurrence, il s'agit de limites très particulières : en effet, dans le cadre du développement d'un logiciel de gestion de Laboratoire d'Analyses Médicales, nous devions pouvoir échantillonner directement des résultats produits par des automates reliés à l'ordinateur via des ports de transmission en série (AUX dans le dialecte MS-DOS). Le problème venait de ce que certains automates très évolués fonctionnent en mode conversationnel. Certes, nous pouvions le résoudre avec DATAFLEX mais au prix d'une lourdeur extrême et d'une notable baisse de performance: le risque à courir était la perte d'informations provenant en particulier d'un automate ignorant tout type d'interruption (« handshake »). La solution nous est vite apparue comme devant être écrite en assembleur de manière à disposer du moyen et le plus compact et le plus rapide.

Le hic: cela faisait plus de 10 ans que je n'avais pas écrit en assembleur ! Courage, lisons : pour se remettre dans le bain, une documentation très complète, bien écrite, avec un vocabulaire simple et propre, en anglais et, à titre d'exemple, les superbes « sources » fournies avec le Concurrent DOS Builder Kit.

La multiplicité entraîne les collisions

L'environnement: multi-tâches, multi-postes, 9 ports série, dont 3 attachés aux automates. Les collisions: lancer de deux postes différents la lecture d'un automate et risquer d'alimenter le même fichier ASCII séquentiel standard.

Les primitives de Concurrent DOS permettent de régler ces sordides questions avec élégance, simplicité et compacité.

J'eusse pu utiliser pour les actions non multi les primitives DOS que Concurrent gère via le vecteur d'interruption 021h (33) mais il est préférable de ne pas mélanger les fonctions gérées par ce vecteur et celles, provenant de CP/M-86 et MP/M-86, gérées via le vecteur OEOh (224).

Enfin, redécouvrant l'assembleur, je me suis astreint à une écriture légendée et pour mon usage et pour celui de mes jeunes collaborateurs qui

n'avaient jamais goûté à ce sport. L'un d'eux, s'étant récemment mis au langage C s'est étonné de l'agrément de l'écriture en assembleur et de sa lisibilité immédiate. J'avoue, pour ma part, avoir éprouvé un vif plaisir à ce retour aux sources (comme l'on dit dans les milieux non informatiques !)

Le programme ci-joint a été développé en tant que maquette d'évaluation. Il ne s'interrompt pas de lui-même en fin de transmission: l'un des automates n'envoie aucun signal de fin de transmission ou de fin de fichier. Nous avons en outre testé ce programme en transmission de données entre ordinateurs incompatibles quant aux supports et quant aux logiciels (un ZENITH Z100 et un HP Vectra 386 RS 20). Nous avons récupéré ainsi environ 20 Mo de données sans erreur à la vitesse de 19400 bauds préréglée par les systèmes respectifs. Pour transmettre en binaire, il faut modifier ce programme en remplaçant l'écho du caractère reçu par une *, par exemple. Le défaut de caractères accentués dans la messagerie est indispensable: toute console n'interprète pas obligatoirement la table ASCII étendue aux caractères internationaux, et peut même prendre certaines valeurs comme des commandes internes de réglage ou de comportement: bonjour le japonais intense ! Toutefois, pour l'agrément de la lecture, j'ai repris le source dans ses commentaires. Nonobstant cette remarque, il est identique à l'original.

Les noms des constantes contenant les n° des primitives invoquées sont tirés du manuel GUIDE DU PROGRAMMEUR fourni avec Concurrent DOS Builder Kit.

La compilation se fait en deux temps:

RASM86 CAPTAUX
LINKCMD CAPTAUX
(produit un fichier exécutable CMD)

A de prochaines retrouvailles en assembleur ou en Dataflex !

Siège social :
36, cours du Chapeau-Rouge
33000 BORDEAUX
Tél. 56 44 21 11 - Fax 56 51 73 94

LOGINE

Direction commerciale :
14, rue Plaisance
75014 PARIS
Tél. (1) 45 39 66 55 - Fax (1) 45 41 47 96

NOUVEAUX

FRENCH ASSISTANT

LE SEUL LOGICIEL CAPABLE DE TRADUIRE UN TEXTE

French Assistant est le seul logiciel d'assistance à la traduction du marché. Outre son dictionnaire bilingue, de plus de 40 000 points d'entrée, et de son conjugueur accessible à tout moment à partir de votre application, il possède un traducteur très puissant qui permet de traduire des textes Anglais.
Prix de lancement : **895 F** HT.

DR MULTIUSER DOS 5.0

LE VRAI MULTI-TACHES

DR MultiUser DOS 5.0 est un système d'exploitation très puissant, multi-tâches. Il exécute les applications DOS sur les micro-ordinateurs à base de processeurs Intel i486, i386 et compatibles.

DR MultiUser DOS supporte la plupart des fonctions réseaux.

DR MultiUser DOS exécute la plupart des applications DOS dont MS-Excel, MS-Windows, WordPerfect, Paradox Lan, Quatro Pro, SideKick, dBase IV Lan, AutoCad, etc. DR MultiUser DOS est une solution DOS multi-complète.
Prix : **5 800 F** HT.

Pour tous ceux qui ont déjà un Concurrent-DOS, nous livrons DR MU-DOS en tant que mise à jour.
Prix : **1 950 F** HT (*nous consulter pour modalités*).

Les utilitaires

Sauvegarde des originaux :	
• **Copy Write (mars 91) (*) - CW**	F HT **895**
Editeur de supports magnétiques :	
• **Disk Explorer (*) - DX**	F HT **895**
Analysez le déroulement de vos programmes en clair :	
• **Analyzer (*) - AII**	F HT **1 695**
Protégez-vous contre les virus :	
• **Antidote (*)**	F HT **745**
La boîte à outils indispensable à vos disques :	
• **Norton Utilities 5.0, en VO, avec disquette traduction**	F HT **995**
L'utilitaire qu'on ne présente plus :	
• **PC Tools Deluxe 6**	F HT **1 350**
L'outil de sauvegarde d'originaux le plus puissant :	
• **Option Board Deluxe 5.4 (*)**	F HT **1 595**
Une compilation d'une multitude de programmes Windows :	
• **101 Windows**	F HT **495**
Outil de diagnostics de votre PC :	
• **QA Plus**	F HT **1 395**
Le seul DOS qui libère 620 Ko de mémoire conventionnelle :	
• **DR DOS 5.0**	F HT **990**
(*) logiciels livrés en version originale, accompagnés d'un manuel en français	
Packs économiques :	
• **Quaid Pack Professionnel (DX + AII)**	F HT **2 072**
• **Quaid Pack Expert (CW + DX + AII)**	F HT **2 788**
• **Mise à jour Copy Write pour les possesseurs d'une ancienne version**	F HT **395**

Prix vacances : le traitement de texte (compatible Multi-tâches et Multi-postes DOS et réseau local NOVELL) Wordcraft Elite (en français) F HT **1 995**

Prix valables jusqu'au 30 septembre 1991

BON DE COMMANDE

Sté .....
Nom .....
Adresse .....
Date ..... CP..... Ville ..... Tél. .....

Veuillez me faire parvenir les produits suivants :

Désignation	*P.U. HT*	*Qté*	*Montant HT*
.....	.....	x	= .....
.....	.....	x	= .....

Participation aux frais de port 40,00
Total HT .....
TVA à 18.6 % .....
Total TTC .....

Signature
.....

❑ Catalogue : merci de me faire parvenir votre catalogue

Je joins à ma commande mon règlement TTC de.....

❑ par chèque bancaire ci-joint

❑ par contre-remboursement + 40 F TTC

❑ par carte bleue n° |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Date d'expiration.....

DEBUG magazine

CAHIER

LISTINGS

SOMMAIRE

- Windows 3 32
- Récursivité 33
- Teapot 34
- Redécouverte ASM 38
- Un noyau multitâches en C++ 41
- Les bibliothèques de liaisons dynamiques DDL sous Windows 44

WINDOWS 3

```
# Standard Windows make file.  The utility MAKE.EXE compares the
# creation date of the file to the left of the colon with the file(s)
# to the right of the colon.  If the file(s) on the right are newer
# then the file on the left, Make will execute all of the command lines
# following this line that are indented by at least one tab or space.
# Any valid MS-DOS command line may be used.

.c.obj:
          $(CC) $<

.rc.res:
          rc -r $<

!if $d(MSC)
CC=cl -c -AS -Gsw -Oas -Zpe
LIB=libw slibce
!else
CC=ztc -c -a -mm -W -b
LIB=libw mlibcew zlm plm
!endif

generic.exe: generic.obj generic.def generic.res
    link4 /NOD generic,,,$(LIB), generic.def
    rc generic.res

generic.res: generic.rc generic.h

generic.obj: generic.c generic.h
```

```
#include <windows.h>             /* requis pour Windows */
#include "generic.h"             /* specifique à cette application */

HANDLE    hInst;                 /* instance courante             */
HBITMAP hTrash;
HANDLE    hAccel;

int PASCAL WinMain(hInstance,hPrevInstance,lpCmdLine,nCmdShow)
HANDLE hInstance;                /* instance courante         */
HANDLE hPrevInstance;            /* instance précédente       */
LPSTR lpCmdLine;                 /* ligne de commande         */
int nCmdShow;                    /* forme d'affichage fenêtre */
{
          MSG  msg;
          HWND hWnd;

          if(!hPrevInstance)          /* d'autres instances en cours ? */
          if(!InitApplication(hInstance)) /* initialisations communes  */
                    return(FALSE);       /* quitte si erreur          */

          /* initialisations spécifiques à cette instance              */
          if(!(hWnd=InitInstance(hInstance,nCmdShow)))
          return(FALSE);

          /* Lit et envoie les messages jusqu'à reception 17-Feb-1991de WM_QUIT */
          while(GetMessage(&msg,NULL,0,0))
          {
          if(!TranslateAccelerator(hWnd,hAccel,&msg))
          {
          TranslateMessage(&msg); /* traduit les codes touches virtuels */
          DispatchMessage(&msg);  /* envoie les messages aux fenêtres   */
          }
          }
          return(msg.wParam);
}

BOOL InitApplication(hInstance)
HANDLE hInstance;
{
          WNDCLASS  wc;

          /* définition de la classe de la fenêtre principale */
          wc.style=NULL;                   /* style                        */
          wc.lpfnWndProc=MainWndProc;      /* fonction de gestion associée */
          wc.cbClsExtra=0;
          wc.cbWndExtra=0;
          wc.hInstance=hInstance;      /* application qui possède la classe */
          wc.hIcon=LoadIcon(hInstance,"GenericIcon");
          wc.hCursor=LoadCursor(NULL,IDC_ARROW);
          wc.hbrBackground=GetStockObject(WHITE_BRUSH);
          wc.lpszMenuName="GenericMenu";
          wc.lpszClassName="GenericWClass";

          /* enregistrement de la classe */
          return (RegisterClass(&wc));
}

BOOL InitInstance(hInstance,nCmdShow)
HANDLE hInstance;
int    nCmdShow;
{
          HWND hWnd;

          /* sauvegarder le vecteur sur l'instance courante */
          hInst=hInstance;

          /* Création de la fenêtre principale */
          hWnd=CreateWindow("GenericWClass","Demo",WS_OVERLAPPEDWINDOW,
          CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,
          NULL,NULL,hInstance,NULL);
          if(!hWnd)
          return(NULL);

          /* chargement des accélérateurs */
          hAccel=LoadAccelerators(hInstance,"GenericAcc");

          /* chargement de l'image bitmap en mémoire */
          hTrash=LoadBitmap(hInstance,"Trash");

          /* le graphique bitmap est ajouté au menu */
          ModifyMenu(GetMenu(hWnd),IDM_TRASH,MF_BYCOMMAND|MF_BITMAP,
                                   IDM_TRASH,(LPSTR)MAKELONG(hTrash,0));

          /* affichage de la fenêtre */
          ShowWindow(hWnd,nCmdShow);
          UpdateWindow(hWnd);
          return(hWnd);
}

long FAR PASCAL MainWndProc(hWnd,message,wParam,lParam)
HWND hWnd;
unsigned message;
WORD wParam;
LONG lParam;
{
          FARPROC lpProcAbout; /* pointeur sur la fonction "About" */
          HMENU   hPopUp;
          POINT   ptMousePos;

          switch(message)
          {
          case WM_COMMAND:       /* message: command from application menu */
                    if(wParam==IDM_ABOUT)
                    {
                    lpProcAbout = MakeProcInstance(About,hInst);
                    DialogBox(hInst,"AboutBox",hWnd,lpProcAbout);
                    FreeProcInstance(lpProcAbout);
                    break;
                    }
                    else
                    return(DefWindowProc(hWnd,message,wParam,lParam));
          case WM_RBUTTONDOWN:
                    hPopUp=GetSubMenu(GetMenu(hWnd),1);
                    if(!hPopUp)
                    {
                    MessageBox(hWnd,"Ce menu n'existe pas","Demo",IDOK);
                    break;
                    }
                    ptMousePos.x=LOWORD(lParam);
                    ptMousePos.y=HIWORD(lParam);
                    ClientToScreen(hWnd,(LPPOINT)&ptMousePos);

if(!TrackPopupMenu(hPopUp,0,ptMousePos.x,ptMousePos.y,0,hWnd,NULL))
MessageBox(hWnd,"Impossible d'afficher ce menu","Demo",IDOK);
                    break;
          case WM_DESTROY:
                    PostQuitMessage(0);
                    break;
          default:
                    return(DefWindowProc(hWnd,message,wParam,lParam));
          }
          return(NULL);
}

BOOL FAR PASCAL About(hDlg,message,wParam,lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
          switch(message)
          {
          case WM_INITDIALOG:
                    return (TRUE);
          case WM_COMMAND:
                    if (wParam==IDOK||wParam==IDCANCEL)
                    {
                    EndDialog(hDlg,TRUE);
                    return(TRUE);
                    }
                    break;
          }
          return(FALSE);
}
```

```
; module-definition file for generic — used by LINK.EXE

NAME           Generic           ; application's module name

DESCRIPTION  'Sample Microsoft Windows Application'
```

```
EXETYPE       WINDOWS    ; required for all Windows applications

STUB           'WINSTUB.EXE' ; Generates error message if application
                             ; is run without Windows

;CODE can be moved in memory and discarded/reloaded
CODE  PRELOAD MOVEABLE DISCARDABLE

;DATA must be MULTIPLE if program can be invoked more than once
DATA  PRELOAD MOVEABLE MULTIPLE


HEAPSIZE     1024
STACKSIZE    5120       ; recommended minimum for Windows applications


; All functions that will be called by any Windows routine
; MUST be exported.

EXPORTS
          MainWndProc   @1   ; name of window processing function
          About           @2   ; name of "About" processing function
```

```
#define IDM_ABOUT 100
#define IDM_TRASH 200

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
BOOL InitApplication(HANDLE);
BOOL InitInstance(HANDLE, int);
long FAR PASCAL MainWndProc(HWND, unsigned, WORD, LONG);
BOOL FAR PASCAL About(HWND, unsigned, WORD, LONG);
```

```
#include "windows.h"
#include "generic.h"

GenericMenu MENU
BEGIN
  POPUP        "&Fichier"
    BEGIN
      MENUITEM "&Effacer",IDM_TRASH
      MENUITEM "A &propos...",IDM_ABOUT
    END
  POPUP        "&Ceci"
    BEGIN
      MENUITEM          "&Est",IDM_ABOUT+1
      MENUITEM          "&Une",IDM_ABOUT+2
      POPUP             "&SΔrie de"
          BEGIN
            MENUITEM     "&Menus",IDM_ABOUT+3
            MENUITEM     "&EmboitΔs",IDM_ABOUT+4
            POPUP        "Pour &former"
              BEGIN
                MENUITEM          "&Toute",IDM_ABOUT+5
                MENUITEM          "&Une",IDM_ABOUT+6
                MENUITEM          "&Cascade",IDM_ABOUT+7
                POPUP             "&De"
                    BEGIN
                      MENUITEM     "&Sous",IDM_ABOUT+8
                      MENUITEM     "&Menus",IDM_ABOUT+9
                    END
              END
          END
    END
END

GenericAcc ACCELERATORS
  BEGIN
    VK_F3,IDM_ABOUT,VIRTKEY
  END

GenericIcon ICON GENERIC.ICO

Trash BITMAP  TRASH.BMP

AboutBox DIALOG 22, 17, 144, 75
STYLE DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
CAPTION "About Generic"
BEGIN
  CTEXT "Microsoft Windows"   -1,  0,  5, 144,      8
  CTEXT "Generic Application" -1,  0, 14, 144,      8
  CTEXT "Version 3.0"                -1,  0, 34, 144,         8
  DEFPUSHBUTTON "OK"     IDOK, 53, 59,  32, 14, WS_GROUP
END
```

RECURSIVITE

```
{*********************************************************************}
{*                                                                   *}
{*  RECURSIVITE appliquée à des fonctions mathématiques classiques   *}
{*                                                                   *}
{*  (c) Jean-Christophe RIAT                          Novembre 1990  *}
{*                                                                   *}
{*  Turbo-Pascal versions 3,4 et 5                                   *}
{*                                                                   *}
{*********************************************************************}

PROGRAM Fonctions_Recursives;


USES Crt;  { A supprimer avec la version 3 de Turbo-Pascal }


VAR choix : CHAR;
    n1,n2 : INTEGER;



{-------------------------------------------------------------------}
{  Fonction récursive de calcul de la factorielle d'un entier n: n!  }
{                                                                   }
{  Utilisation de la relation: n! = n * (n-1)!                      }
{  et de la condition d'arrêt: 0! = 1                               }
{-------------------------------------------------------------------}
FUNCTION Factorielle(n : INTEGER) : REAL;

BEGIN
  IF n=0
    THEN
      Factorielle:=1
    ELSE
      Factorielle:=n*Factorielle(n-1);
END;



{-------------------------------------------------------------------}
{  Fonction récursive de calcul du nombre de combinaisons possibles  }
{  de p éléments pris parmi n: Cn,p                                 }
{                                                                   }
{  Utilisation de la relation: Cn,p = Cn-1,p + Cn-1,p-1              }
{  et des conditions d'arrêt : Cn,1 = n                             }
{                              Cn,n = 1                             }
{-------------------------------------------------------------------}
FUNCTION Cnp(n,p : INTEGER) : REAL;

BEGIN
  IF p=1
    THEN
      Cnp:=n
    ELSE
      IF p=n
        THEN
          Cnp:=1
        ELSE
          Cnp:=Cnp(n-1,p)+Cnp(n-1,p-1);
END;


{-------------------------------------------------------------------}
{  Fonction récursive de calcul du nombre d'arrangements possibles   }
{  de p éléments pris parmi n: An,p                                 }
{                                                                   }
{  Utilisation des relations:  An,p = An-1,p + p*An-1,p-1            }
{                              An,n = n * An-1,p-1                  }
{  et de la condition d'arrêt: An,1 = n                             }
{-------------------------------------------------------------------}
FUNCTION Anp(n,p : INTEGER) : REAL;

BEGIN
  IF p=1
    THEN
      Anp:=n
    ELSE
      IF p=n
        THEN
          Anp:=n*Anp(n-1,p-1)
        ELSE
          Anp:=Anp(n-1,p)+p*Anp(n-1,p-1);
END;


{-------------------------------------------------------------------}
{  Fonction récursive de calcul du PGCD de deux entiers a et b      }
{                                                                   }
{  Utilisation des relations:  PGCD(a,b) = PGCD(b,a)                }
{                              si a>b,PGCD(a,b) = PGCD(a MOD b,b)   }
{  et de la condition d'arrêt: PGCD(a,0) = a                        }
{-------------------------------------------------------------------}
```

```
FUNCTION PGCD(a,b : INTEGER) : LONGINT;

BEGIN
  IF a<b
    THEN
      PGCD:=PGCD(b,a)
    ELSE
      IF b=0
        THEN
          PGCD:=a
        ELSE
          PGCD:=PGCD(a MOD b,b);
END;


{-------------------------------------------------------------------}
{  Fonction récursive de calcul du PPCM de deux entiers a et b      }
{                                                                   }
{  Utilisation des relations:  PPCM(a,b) = PPCM(b,a)                }
{                              si a>b,PPCM(a,b) = a/(a-b)*PPCM(a-b,b)}
{  et de la condition d'arrêt: PPCM(a,a)=a                          }
{-------------------------------------------------------------------}
FUNCTION PPCM(a,b : INTEGER) : LONGINT;

BEGIN
  IF a<b
    THEN
      PPCM:=PPCM(b,a)
    ELSE
      IF a=b
        THEN
          PPCM:=a
        ELSE
          PPCM:=ROUND(a/(a-b)*PPCM(a-b,b));
END;


{-------------------------------------------------------------------}
{ Procédure de lecture d'un nombre entier positif                   }
{-------------------------------------------------------------------}
PROCEDURE Lire_Nombre(VAR nombre : INTEGER; texte : STRING);

BEGIN
  REPEAT
    WRITE('Entrer ',texte,': ');
    READLN(nombre);
    IF nombre<0
      THEN
        WRITELN('Le nombre doit être positif!!!');
  UNTIL nombre>=0;
END;



{ Le corps du programme principal... }
BEGIN
  REPEAT
    ClrScr;
    WRITELN('1. Calcul de factorielle: n!');
    WRITELN('2. Calcul de combinaison: Cnp');
    WRITELN('3. Calcul d''arrangement : Anp');
    WRITELN('4. Calcul du PGCD de 2 entiers: PGCD(a,b)');
    WRITELN('5. Calcul du PPCM de 2 entiers: PPCM(a,b)');
    WRITELN('6. Fin');
    WRITELN;
    REPEAT
      WRITE('Votre choix: ':30);
      READLN(choix);
    WRITELN;
    UNTIL choix IN ['1'..'6'];
    CASE choix OF
      '1' : BEGIN
              Lire_Nombre(n1,'la factorielle à calculer');
              WRITELN(n1,'!=',Factorielle(n1):0:0);
            END;
      '2' : BEGIN
              REPEAT
                Lire_Nombre(n1,'la valeur de n');
                Lire_Nombre(n2,'la valeur de p');
                IF n2>n1
                  THEN
                    WRITELN('ATTENTION: n doit être supérieur à p');
              UNTIL n1>=n2;
              WRITELN('Cnp(',n1,',',n2,')=',Cnp(n1,n2):0:0);
            END;
      '3' : BEGIN
              REPEAT
                Lire_Nombre(n1,'la valeur de n');
                Lire_Nombre(n2,'la valeur de p');
                IF n2>n1
                  THEN
                    WRITELN('ATTENTION: n doit être supérieur à p');
              UNTIL n1>=n2;
              WRITELN('Anp(',n1,',',n2,')=',Anp(n1,n2):0:0);
            END;
      '4' : BEGIN
              Lire_Nombre(n1,'valeur du premier nombre');
              Lire_Nombre(n2,'valeur du second nombre');
              WRITELN('PGCD(',n1,',',n2,')=',PGCD(n1,n2));
            END;
      '5' : BEGIN
              Lire_Nombre(n1,'valeur du premier nombre');
              Lire_Nombre(n2,'valeur du second nombre');
              WRITELN('PPCM(',n1,',',n2,')=',PPCM(n1,n2));
            END;
      '6' :
    END;
    WRITELN;
    WRITELN('Appuyer une touche quelconque...':60);
    WHILE ReadKey='' DO;
  UNTIL choix='6';
  WRITELN;
  WRITELN('Enchanté d''avoir travaillé avec vous. Au revoir...');
END.
```

TEAPOT

```
/*
          Programme           : TEAPOT.C
          Fonction            : Faire tourner une théière dans l'espace puis
                                la tracer avec des surfaces Bezier.
          Date                : 19/02/91
          Developpement       : JP-COTE
          Compilateur         : TURBO-C version 2.0
          programme compilé en modèle LARGE
*/

#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>
#include "teapot.h"

PRIVATE POINT3D pole[] = {
                                        #include "c:\points.dat"
                                        };
PRIVATE int          patch[] = {
                                        #include "c:\patch.dat"
                                        };

          int       trans_x, trans_y, inc_trx, inc_try;
          float     rho, theta, phi;
          float     dist_ecran;
          float     inc_angle, inc_ecran;
          float     aux1, aux2, aux3, aux4;
          float     aux5, aux6, aux7, aux8;

/* Les fichiers *.BGI sont supposés être dans le répertoire
          C:\TC
*/
PRIVATE void Init_Graphics( void )
{
          int       g_Driver, g_Mode, g_Error;
          char      *path_to_driver = "C:\\TC";

#ifdef _EGA
          g_Driver = EGAMONOHI;
          g_Mode   = 0;
#endif

#ifdef _MONO
          g_Driver = HERCMONO;
          g_Mode = HERCMONOHI;
#endif

#ifdef _VGA
          g_Driver = VGA;
          g_Mode   = VGAHI;
#endif

          initgraph( &g_Driver, &g_Mode, path_to_driver );
          if ( ( g_Error = graphresult() ) < 0 )
          {
                    printf("\nErreur d'init graphique : g_Error = %d\n", g_Error );
                    exit( 1 );
          }
} /* End of INIT_GRAPHICS */

PRIVATE void Init_Data( void )
{
          rho = 5.0, theta = 270.0, phi = -20.0;
          inc_angle  = 30.0, inc_ecran  = 40, dist_ecran = 300.0;
          trans_x = 0, trans_y = 100;
          inc_trx = 30, inc_try = 30;

} /* End of INIT_DATA */

PRIVATE void Init_Projection( float theta, float phi )
{
          float     th, ph;

          th = PI * ( theta/180.0 );
          ph = PI * ( phi/180.0 );
          aux1 = sin( th );
          aux2 = sin( ph );
          aux3 = cos( th );
          aux4 = cos( ph );
          aux5 = aux3 * aux2;
          aux6 = aux1 * aux2;
          aux7 = aux3 * aux4;
          aux8 = aux1 * aux4;
```

```
} /* End of INIT_PROJECTION */

PRIVATE int C_X( float x )
{
        return( (int) ( (x + trans_x)* ECHELLE + (BASECRAN/2.0) ) );
} /* End of C_X */

PRIVATE int C_Y( float y )
{
        return( (int) ( (HAUTECRAN/2.0) - y + trans_y ) );
} /* End of C_Y */

/* La matrice Bezier [Mb] est de la forme :

                        base[0]  base[1]  base[2]  base[3]
                [Mb] =  base[4]  base[5]  base[6]  base[7]
                        base[8]  base[9]  base[10] base[11]
                        base[12] base[13] base[14] base[15]

*/
PRIVATE void Base_Bezier( float *base )
{
        base[0]  = -1.0, base[1] = 3.0, base[2]  = -3.0, base[3] = 1.0;
        base[4]  = 3.0, base[5]  = -6.0, base[6] = 3.0, base[7]  = 0.0;
        base[8]  = -3.0, base[9] = 3.0, base[10] = 0.0, base[11] = 0.0;
        base[12] = 1.0, base[13] = 0.0, base[14] = 0.0, base[15] = 0.0;
} /* End of BASE_BEZIER */

/* L'évaluation se fait par :
                                    mat[0]  mat[1]  mat[2]  mat[3]
                [ u3, u2, u, 1] *   mat[4]  mat[5]  mat[6]  mat[7]
                                    mat[8]  mat[9]  mat[10] mat[11]
                                    mat[12] mat[13] mat[14] mat[15]
        Le résultat est stocké sous la forme :
                        (tmp[3], tmp[2], tmp[1], tmp[0])
        u est le paramêtre d'évaluation
*/
PRIVATE void Horner_U( float *mat, float u, float *tmp )
{
        tmp[3] = mat[12] + u * ( mat[8]  + u * ( mat[4] + u * mat[0] ) );
        tmp[2] = mat[13] + u * ( mat[9]  + u * ( mat[5] + u * mat[1] ) );
        tmp[1] = mat[14] + u * ( mat[10] + u * ( mat[6] + u * mat[2] ) );
        tmp[0] = mat[15] + u * ( mat[11] + u * ( mat[7] + u * mat[3] ) );
} /* End of HORNER_U */

/* L'évaluation se fait par :
        mat[0]  mat[1]  mat[2]  mat[3]          [v3]
        mat[4]  mat[5]  mat[6]  mat[7] *        [v2]
        mat[8]  mat[9]  mat[10] mat[11]         [v]
        mat[12] mat[13] mat[14] mat[15]         [1]


                                                                t
        Le résultat produit [ tmp[3], tmp[2], tmp[1], tmp[0] ]
*/
PRIVATE void Horner_V( float *mat, float v, float *tmp )
{
        tmp[3] = mat[3] +  v * ( mat[2]  + v * ( mat[1] +  v * mat[0] ) );
        tmp[2] = mat[7] +  v * ( mat[6]  + v * ( mat[5] +  v * mat[4] ) );
        tmp[1] = mat[11] + v * ( mat[10] + v * ( mat[9] +  v * mat[8] ) );
        tmp[0] = mat[15] + v * ( mat[14] + v * ( mat[13] + v * mat[12] ) );
} /* End of HORNER_V */

/* On a le résultat suivant :
                        tbase[0]  tbase[4]  tbase[8]  tbase[12]
        [ Mtb] =        tbase[1]  tbase[5]  tbase[9]  tbase[13]
                        tbase[2]  tbase[6]  tbase[10] tbase[14]
                        tbase[3]  tbase[7]  tbase[11] tbase[15]
*/
PRIVATE void Base_Bezier_Transp( float *base, float *tbase )
{
#define   VAL       15
#define NBPTS       4
        register int        i;

        for ( tbase[0] = base[0], i = 1; i < VAL; i++ )
        {
                register int        sum;

                sum = ( NBPTS * i ) % VAL;
                tbase[i]  = base[sum];
        }
        tbase[VAL] = base[VAL];
}  /* End of BASE_BEZIER_TRANSP */


PRIVATE void Matrice_Mul( float *A, float *B, float *C )
{
#define NBPTS       4
        register int        i;

        /* Lignes de la première matrice [ A ] */
        for ( i = 0; i < NBPTS; i++ )
        {
                register int        j, k;

                /* Colonnes de la seconde matrice [ B ] */
                for ( j = 0; j < NBPTS; j++ )
                {
                        float     sum;

                /* Affectation de la matrice résultat */
                for ( sum = 0.0, k = 0; k < NBPTS; k++ )
                {
                        sum += A[i * NBPTS + k] * B[k * NBPTS + j];
                        C[i * NBPTS + j] = sum;
                }
                }
        }
}       /* End of MATRICE_MUL */

PRIVATE void Transform_3d_2d( float xp, float yp, float zp, float
                *x_proj, float *y_proj )
{
        float     x_obs, y_obs, z_obs;

        x_obs = -xp * aux1 + yp * aux3;
        y_obs = -xp * aux5 - yp * aux6 + zp * aux4;
        z_obs = -xp * aux7 - yp * aux8 - zp * aux2 + rho;
        *x_proj = dist_ecran * x_obs/z_obs;
        *y_proj = dist_ecran * y_obs/z_obs;
} /* End of TRANSFORM_3d_2D */

PRIVATE void Read_Points( float *xx, float *yy, float *zz, int num )
{
        register int        i;

        /* Pour le patch NUM, lecture des points du B-Reseau */
        for ( i = 0; i < 16; i++ )
        {
                xx[i] = pole[ patch[ num * 16 + i ] ].x;
                yy[i] = pole[ patch[ num * 16 + i ] ].y;
                zz[i] = pole[ patch[ num * 16 + i ] ].z;
        }
} /* End of READ_POINTS */

PRIVATE void Get_Vecteur( POINT3D *vect, float *xx, float *yy, float
                *zz, float u )
{
        vect->x = xx[0] + u * ( xx[1] + u * ( xx[2] + u * xx[3] ) );
        vect->y = yy[0] + u * ( yy[1] + u * ( yy[2] + u * yy[3] ) );
        vect->z = zz[0] + u * ( zz[1] + u * ( zz[2] + u * zz[3] ) );
} /* End of GET_VECTEUR */

PRIVATE void Patch_Bezier( int inc_N, int inc_M )
{
        register int        i, num;
        float               x_proj, y_proj;
        float               base[16], tbase[16], tmp[16];
        /*
                Récupération des coefficients des 2 matrices dans
                les tableaux BASE et TBASE
        */
        Base_Bezier( base );
        Base_Bezier_Transp( base, tbase );
        for ( num = 0; num < 32; num++ )
        {
                float     ptx[16], pty[16], ptz[16];
                float     Matx[16], Maty[16], Matz[16];
                float     xx[16], yy[16], zz[16];
                Read_Points( ptx, pty, ptz, num );
                /* Pour chaque composante on éffectue le produit
                        [ Matx ] = [ M ] * [ Pijx ] * [ Mt ]
                */
                Matrice_Mul( base, ptx, tmp );
                Matrice_Mul( tmp, tbase, Matx );
                /*
                [ Maty ] = [ M ] * [ Pijy ] * [ Mt ]
                */
                Matrice_Mul( base, pty, tmp );
                Matrice_Mul( tmp, tbase, Maty );
                /*
                        [ Matz ] = [ M ] * [ Pijz ] * [ Mt ]
                */
                Matrice_Mul( base, ptz, tmp );
                Matrice_Mul( tmp, tbase, Matz );
                /* Trace suivant les ISO_V */
                for ( i = 0; i <= inc_N; i++ )
                {
                        register int        j;
                        float               u;

                        u = (float) i/inc_N;
                        Horner_U( Matx, u, xx );
                        Horner_U( Maty, u, yy );
                        Horner_U( Matz, u, zz );
                        Transform_3d_2d(xx[0],yy[0], zz[0], &x_proj,&y_proj);
                        moveto( C_X( x_proj ), C_Y( y_proj ) );
                        for ( j = 1; j <= inc_M; j++ )
                        {
                                float     v;
                                POINT3D   vect;

                                v = (float) j/inc_M;
                                Get_Vecteur( &vect, xx, yy, zz, v );
                                Transform_3d_2d(vect.x, vect.y, vect.z,
                                &x_proj, &y_proj);
                                lineto( C_X( x_proj ), C_Y( y_proj )
);
                        }
                } /* End FOR */
                /* Trace suivant les ISO_U */
                for ( i = 0; i <= inc_N; i++ )
                {
                        register int        j;
                        float               v;

                        v = (float)i/inc_M;
                        Horner_V( Matx, v, xx );
                        Horner_V( Maty, v, yy );
                        Horner_V( Matz, v, zz );
                        Transform_3d_2d( xx[0], yy[0], zz[0],
```

```
                        &x_proj, &y_proj );
                        moveto( C_X( x_proj ), C_Y( y_proj ) );
                        for ( j = 1; j <= inc_N; j++ )
                        {
                        float               u;
                        POINT3D    vect;

                        u = (float)j/inc_N;
                        Get_Vecteur( &vect, xx, yy, zz, u );
                        Transform_3d_2d(vect.x, vect.y, vect.z,&x_proj,
                        &y_proj);
                        lineto( C_X( x_proj ), C_Y( y_proj ) );
                }
                }       /* End of FOR de i */

        } /* End of FOR de num */
} /* End of PATCH_BEZIER */

/*
        La numérotation est la suivante :
                P0      P1      P2      P3
                P4      P5      P6      P7
                P8      P9      P10     P11
                P12     P13     P14     P15
*/
PRIVATE void Points_Controle( float theta, float phi, int couleur )
{
        register int        num;

        setcolor( couleur );
        Init_Projection( theta, phi );
        /* Pour chaque patch */
        for ( num = 0; num < 32; num++ )
        {
                register int        i;
                float               x_proj, y_proj;
                float               xx[16], yy[16], zz[16];

                Read_Points( xx, yy, zz, num );
                /* Tracé suivant les ISO_U */
                for ( i = 0 ; i < 4 ; i++ )
                {
                        register int        j;
                        int                 tp;

                        tp = (4 * i);
                        Transform_3d_2d(xx[tp],yy[tp],zz[tp],&x_proj,&y_proj);
                        moveto( C_X( x_proj ), C_Y( y_proj ) );
                        for ( j = (tp + 1); j < 4 * (i + 1) ; j++ )
                        {
                        Transform_3d_2d(xx[j],yy[j],zz[j],&x_proj,&y_proj);
                        lineto( C_X( x_proj ), C_Y( y_proj ) );
                        }
                }
                /* Tracé suivant les ISO_V */
                for ( i = 0 ; i < 4 ; i++ )
                {
                register int        j;

                Transform_3d_2d(xx[i], yy[i], zz[i], &x_proj, &y_proj);
                moveto( C_X( x_proj ), C_Y( y_proj ) );
                for ( j = i; j <= (i + 12); j += 4 )
                {
                        Transform_3d_2d(xx[j],yy[j],zz[j],&x_proj, &y_proj);
                        lineto( C_X( x_proj ), C_Y( y_proj ) );
                }
                }

        } /* End of FOR de num */
} /* End of POINTS_CONTROLE */


PRIVATE void Rotation_Teapot( int couleur )
{
        static int appui_touche;

        Points_Controle( theta, phi, couleur );
        while( appui_touche != ESC )
        {
                if ( kbhit() )
                {
                switch( appui_touche = getch() )
                {
                case FLECHE_DROITE: phi        += inc_angle; break;
                case FLECHE_GAUCHE: phi        -= inc_angle; break;
                case FLECHE_HAUTE : theta      += inc_angle; break;
                case FLECHE_BASSE : theta      -= inc_angle; break;
                case PLUS    : dist_ecran      += inc_ecran; break;
                case MOINS        : dist_ecran -= inc_ecran; break;
                case F1           : trans_x    += inc_trx; break;
                case F2           : trans_x    -= inc_trx; break;
                case F3           : trans_y    += inc_try; break;
                case F4           : trans_y    -= inc_try; break;
                case ESC     : continue;
                }
                cleardevice();       /* Efface l'écran */
                Points_Controle( theta, phi, couleur );
                }
        }
} /* End of ROTATION_TEAPOT */

void main( void )
{
        int                 pas_x = 6, pas_y = 6;

        Init_Data();
        Init_Graphics();
        Rotation_Teapot( GREEN );
        cleardevice();
        setcolor( YELLOW );
        Patch_Bezier( pas_x, pas_y );
        getch();
        closegraph();
} /* End of MAIN */
```

```
/*
        Header              : TEAPOT.H
        Date                          : 25/02/91
        Commentaires        : Header du programme TEAPOT.C
*/

/* Suivant votre type d'écran, il faudra adapter la variable
   ci-dessous. */
#define _EGA        /* Variable globale TYPE D'ECRAN */

#ifdef _EGA
#define BASECRAN    639
#define HAUTECRAN   349
#endif

#ifdef _VGA
#define BASECRAN    639
#define HAUTECRAN   479
#endif

#ifdef _MONO
#define BASECRAN    719
#define HAUTECRAN   347
#endif

/* Test d'existence des variables */
#ifndef PUBLIC
#define   PUBLIC
#endif
#ifndef PRIVATE
#define PRIVATE     static
#endif

#define PI                      ( 4.0 * atan( 1.0 ) )
#define ECHELLE                 1.35

/* Définition des touches. */
#define ESC                             27
#defineFLECHE_DROITE 72
#define FLECHE_GAUCHE           80
#define FLECHE_HAUTE 75
#define FLECHE_BASSE 77
#define PLUS                            43
#define MOINS                           45
#define F1                              59
#define F2                              60
#define F3                              61
#define F4                              62

typedef struct {    float x, y, z; } POINT3D;
```

```
/* BORD */
0,  1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
3, 16,17,18, 7,19,20,21,11,22,23,24,15,25,26,27,
18,28,29,30,21,31,32,33,24,34,35,36,27,37,38,39,
30,40,41, 0,33,42,43, 4,36,44,45, 8,39,46,47,12,

/* CORPS */
12, 13, 14, 15, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
15, 25, 26, 27, 51, 60, 61, 62, 55, 63, 64, 65, 59, 66, 67, 68,
27, 37, 38, 39, 62, 69, 70, 71, 65, 72, 73, 74, 68, 75, 76, 77,
39, 46, 47, 12, 71, 78, 79, 48, 74, 80, 81, 52, 77, 82, 83, 56,
56, 57, 58, 59, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
59, 66, 67, 68, 87, 96, 97, 98, 91, 99,100,101, 95,102,103,104,
68, 75, 76, 77, 98,105,106,107,101,108,109,110,104,111,112,113,
77, 82, 83, 56,107,114,115, 84,110,116,117, 88,113,118,119, 92,

/* POIGNEE */
120,121,122,123,124,125,126,127,128,129,130,131,132,133,134,135,
123,136,137,120,127,138,139,124,131,140,141,128,135,142,143,132,
132,133,134,135,144,145,146,147,148,149,150,151, 68,152,153,154,
135,142,143,132,147,155,156,144,151,157,158,148,154,159,160, 68,

/* BEC */
161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,
164,177,178,161,168,179,180,165,172,181,182,169,176,183,184,173,
173,174,175,176,185,186,187,188,189,190,191,192,193,194,195,196,
176,183,184,173,188,197,198,185,192,199,200,189,196,201,202,193,

/* COUVERCLE */
203,203,203,203,206,207,208,209,210,210,210,210,211,212,213,214,
203,203,203,203,209,216,217,218,210,210,210,210,214,219,220,221,
203,203,203,203,218,223,224,225,210,210,210,210,221,226,227,228,
203,203,203,203,225,229,230,206,210,210,210,210,228,231,232,211,
211,212,213,214,233,234,235,236,237,238,239,240,241,242,243,244,
214,219,220,221,236,245,246,247,240,248,249,250,244,251,252,253,
221,226,227,228,247,254,255,256,250,257,258,259,253,260,261,262,
228,231,232,211,256,263,264,233,259,265,266,237,262,267,268,241,
```

```
/* FOND */
269,269,269,269,278,279,280,281,274,275,276,277,270,271,272,273,
269,269,269,269,281,288,289,290,277,285,286,287,273,282,283,284,
269,269,269,269,290,297,298,299,287,294,295,296,284,291,292,293,
269,269,269,269,299,304,305,278,296,302,303,274,293,300,301,270
```

```
1.40000, 0.00000, 2.40000,
1.40000, -.78400, 2.40000,
.784000, -1.4000, 2.40000,
0.00000, -1.4000, 2.40000,
1.33750, 0.00000, 2.53125,
1.33750, -.74900, 2.53125,
.749000, -1.3375, 2.53125,
0.00000, -1.3375, 2.53125,
1.43750, 0.00000, 2.53125,
1.43750, -.80500, 2.53125,
.805000, -1.4375, 2.53125,
0.00000, -1.4375, 2.53125,
1.50000, 0.00000, 2.40000,
1.50000, -.84000, 2.40000,
.840000, -1.5000, 2.40000,
0.00000, -1.5000, 2.40000,      /* 16 */

-.78400, -1.4000, 2.40000,
-1.4000, -.78400, 2.40000,
-1.4000, 0.00000, 2.40000,
-.74900, -1.3375, 2.53125,
-1.3375, -.74900, 2.53125,
-1.3375, 0.00000, 2.53125,
-.80500, -1.4375, 2.53125,
-1.4375, -.80500, 2.53125,
-1.4375, 0.00000, 2.53125,
-.84000, -1.5000, 2.40000,
-1.5000, -0.8400, 2.40000,
-1.5000, 0.00000, 2.40000,
-1.4000, 0.78400, 2.40000,
-.78400, 1.40000, 2.40000,
0.00000, 1.40000, 2.40000,
-1.3375, .749000, 2.53125,      /* 32 */

-.74900, 1.33750, 2.53125,
0.00000, 1.33750, 2.53125,
-1.4375, .805000, 2.53125,
-.80500, 1.43750, 2.53125,
0.00000, 1.43750, 2.53125,
-1.5000, 0.84000, 2.40000,
-0.8400, 1.50000, 2.40000,
0.00000, 1.50000, 2.40000,
.784000, 1.40000, 2.40000,
1.40000, .784000, 2.40000,
.749000, 1.33750, 2.53125,
1.33750, .749000, 2.53125,
.805000, 1.43750, 2.53125,
1.43750, .805000, 2.53125,
.840000, 1.50000, 2.40000,
1.50000, .840000, 2.40000,      /* 48 */

1.75000, 0.00000, 1.87500,
1.75000, -.98000, 1.87500,
.980000, -1.7500, 1.87500,
0.00000, -1.7500, 1.87500,
2.00000, 0.00000, 1.35000,
2.00000, -1.1200, 1.35000,
1.12000, -2.0000, 1.35000,
0.00000, -2.0000, 1.35000,
2.00000, 0.00000, .900000,
2.00000, -1.1200, .900000,
1.12000, -2.0000, .900000,
0.00000, -2.0000, .900000,
-.98000, -1.7500, 1.87500,
-1.7500, -.98000, 1.87500,
-1.7500, 0.00000, 1.87500,
-1.1200, -2.0000, 1.35000,      /* 64 */

-2.0000, -1.1200, 1.35000,
-2.0000, 0.00000, 1.35000,
-1.1200, -2.0000, .900000,
-2.0000, -1.1200, .900000,
-2.0000, 0.00000, .900000,
-1.7500, .980000, 1.87500,
-.98000, 1.75000, 1.87500,
0.00000, 1.75000, 1.87500,
-2.0000, 1.12000, 1.35000,
-1.1200, 2.00000, 1.35000,
0.00000, 2.00000, 1.35000,
-2.0000, 1.12000, .900000,
-1.1200, 2.00000, .900000,
0.00000, 2.00000, .900000,
.980000, 1.75000, 1.87500,
1.75000, .980000, 1.87500,      /* 80 */

1.12000, 2.00000, 1.35000,
2.00000, 1.12000, 1.35000,
1.12000, 2.00000, .900000,
2.00000, 1.12000, .900000,
2.00000, 0.00000, .450000,
2.00000, -1.1200, .450000,
1.12000, -2.0000, .450000,
0.00000, -2.0000, .450000,
1.50000, 0.00000, .225000,
1.50000, -.84000, .225000,
.840000, -1.5000, .225000,
0.00000, -1.5000, .225000,
1.50000, 0.00000, .150000,
1.50000, -.84000, .150000,
.840000, -1.5000, .150000,
0.00000, -1.5000, .150000,      /* 96 */

-1.1200, -2.0000, .450000,
-2.0000, -1.1200, .450000,
-2.0000, 0.00000, .450000,
-.84000, -1.5000, .225000,
-1.5000, -.84000, .225000,
-1.5000, 0.00000, .225000,
-.84000, -1.5000, .150000,
-1.5000, -.84000, .150000,
-1.5000, 0.00000, .150000,
-2.0000, 1.12000, .450000,
-1.1200, 2.00000, .450000,
0.00000, 2.00000, .450000,
-1.5000, .840000, .225000,
-.84000, 1.50000, .225000,
0.00000, 1.50000, .225000,
-1.5000, .840000, .150000,      /* 112 */

-.84000, 1.50000, .150000,
0.00000, 1.50000, .150000,
1.12000, 2.00000, .450000,
2.00000, 1.12000, .450000,
.840000, 1.50000, .225000,
1.50000, .840000, .225000,
.840000, 1.50000, .150000,
1.50000, .840000, .150000,
-1.6000, 0.00000, 2.02500,
-1.6000, -.30000, 2.02500,
-1.5000, -.30000, 2.25000,
-1.5000, 0.00000, 2.25000,
-2.3000, 0.00000, 2.02500,
-2.3000, -.30000, 2.02500,
-2.5000, -.30000, 2.25000,
-2.5000, 0.00000, 2.25000,      /* 128 */

-2.7000, 0.00000, 2.02500,
-2.7000, -.30000, 2.02500,
-3.0000, -.30000, 2.25000,
-3.0000, 0.00000, 2.25000,
-2.7000, 0.00000, 1.80000,
-2.7000, -.30000, 1.80000,
-3.0000, -.30000, 1.80000,
-3.0000, 0.00000, 1.80000,
-1.5000, .300000, 2.25000,
-1.6000, .300000, 2.02500,
-2.5000, .300000, 2.25000,
-2.3000, .300000, 2.02500,
-3.0000, .300000, 2.25000,
-2.7000, .300000, 2.02500,
-3.0000, .300000, 1.80000,
-2.7000, .300000, 1.80000,      /* 144 */

-2.7000, .000000, 1.57500,
-2.7000, -.30000, 1.57500,
-3.0000, -.30000, 1.35000,
-3.0000, .000000, 1.35000,
-2.5000, .000000, 1.12500,
-2.5000, -.30000, 1.12500,
-2.6500, -.30000, .937500,
-2.6500, .000000, .937500,
-2.0000, -.30000, .900000,
-1.9000, -.30000, .600000,
-1.9000, .000000, .600000,
-3.0000, .300000, 1.35000,
-2.7000, .300000, 1.57500,
-2.6500, .300000, .937500,
-2.5000, .300000, 1.12500,
-1.9000, .300000, .600000,      /* 160 */

-2.0000, .300000, .900000,
1.70000, .000000, 1.42500,
1.70000, -.66000, 1.42500,
1.70000, -.66000, .600000,
1.70000, .000000, .600000,
2.60000, .000000, 1.42500,
2.60000, -.66000, 1.42500,
3.10000, -.66000, .825000,
3.10000, .000000, .825000,
2.30000, .000000, 2.10000,
2.30000, -.25000, 2.10000,
2.40000, -.25000, 2.02500,
2.40000, .000000, 2.02500,
2.70000, .000000, 2.40000,
2.70000, -.25000, 2.40000,
3.30000, -.25000, 2.40000,      /* 176 */

3.30000, .000000, 2.40000,
1.70000, .660000, .600000,
1.70000, .660000, 1.42500,
3.10000, .660000, .825000,
2.60000, .660000, 1.42500,
2.40000, .250000, 2.02500,
2.30000, .250000, 2.10000,
3.30000, .250000, 2.40000,
2.70000, .250000, 2.40000,
2.80000, .000000, 2.47500,
2.80000, -.25000, 2.47500,
3.52500, -.25000, 2.49375,
3.52500, .000000, 2.49375,
2.90000, .000000, 2.47500,
```

```
2.90000, -.15000, 2.47500,
3.45000, -.15000, 2.51250,     /* 192 */

3.45000, .000000, 2.51250,
2.80000, .000000, 2.40000,
2.80000, -.15000, 2.40000,
3.20000, -.15000, 2.40000,
3.20000, .000000, 2.40000,
3.52500, .250000, 2.49375,
2.80000, .250000, 2.47500,
3.45000, .150000, 2.51250,
2.90000, .150000, 2.47500,
3.20000, .150000, 2.40000,
2.80000, .150000, 2.40000,
.000000, .000000, 3.15000,
0.00000, -.00200, 3.15000,
0.00200, .000000, 3.15000,
0.80000, 0.00000, 3.15000,
0.80000, -.45000, 3.15000,     /* 208 */

0.45000, -.80000, 3.15000,
0.00000, -.80000, 3.15000,
0.00000, 0.00000, 2.85000,
0.20000, 0.00000, 2.70000,
0.20000, -0.1120, 2.70000,
0.11200, -0.2000, 2.70000,
0.00000, -0.2000, 2.70000,
-0.0020, 0.00000, 3.15000,
-0.4500, -.80000, 3.15000,
-0.8000, -0.4500, 3.15000,
-0.8000, 0.00000, 3.15000,
-0.1120, -0.2000, 2.70000,
-0.2000, -0.1120, 2.70000,
-0.2000, 0.00000, 2.70000,
0.00000, 0.00200, 3.15000,
-0.8000, 0.45000, 3.15000,     /* 224 */

-0.4500, 0.80000, 3.15000,
0.00000, 0.80000, 3.15000,
-0.2000, 0.11200, 2.70000,
-0.1120, 0.20000, 2.70000,
0.00000, 0.20000, 2.70000,
0.45000, 0.80000, 3.15000,
0.80000, 0.45000, 3.15000,
0.11200, 0.20000, 2.70000,
0.20000, 0.11200, 2.70000,
0.40000, 0.00000, 2.55000,
0.40000, -0.2240, 2.55000,
0.22400, -0.4000, 2.55000,
0.00000, -0.4000, 2.55000,
1.30000, 0.00000, 2.55000,
1.30000, -0.7280, 2.55000,
0.72800, -1.3000, 2.55000,     /* 240 */

0.00000, -1.3000, 2.55000,
1.30000, 0.00000, 2.40000,
1.30000, -0.7280, 2.40000,
0.72800, -1.3000, 2.40000,
0.00000, -1.3000, 2.40000,
-0.2240, -0.4000, 2.55000,
-0.4000, -0.2240, 2.55000,
-0.4000, 0.00000, 2.55000,
-0.7280, -1.3000, 2.55000,
-1.3000, -0.7280, 2.55000,
-1.3000, 0.00000, 2.55000,
-0.7280, -1.3000, 2.40000,
-1.3000, -0.7280, 2.40000,
-1.3000, 0.00000, 2.40000,
-0.4000, 0.22400, 2.55000,
-0.2240, 0.40000, 2.55000,     /* 256 */

0.00000, 0.40000, 2.55000,
-1.3000, 0.72800, 2.55000,
-0.7280, 1.30000, 2.55000,
0.00000, 1.30000, 2.55000,
-1.3000, 0.72800, 2.40000,
-0.7280, 1.30000, 2.40000,
0.00000, 1.30000, 2.40000,
0.22400, 0.40000, 2.55000,
0.40000, 0.22400, 2.55000,
0.72800, 1.30000, 2.55000,
1.30000, 0.72800, 2.55000,
0.72800, 1.30000, 2.40000,
1.30000, 0.72800, 2.40000,
0.00000, 0.00000, 0.00000,
1.50000, 0.00000, 0.15000,
1.50000, 0.84000, 0.15000,     /* 272 */

0.84000, 1.50000, 0.15000,
0.00000, 1.50000, 0.15000,
1.50000, 0.00000, 0.07500,
1.50000, 0.84000, 0.07500,
0.84000, 1.50000, 0.07500,
0.00000, 1.50000, 0.07500,
1.42500, 0.00000, 0.00000,
1.42500, 0.79800, 0.00000,
0.79800, 1.42500, 0.00000,
0.00000, 1.42500, 0.00000,
-0.8400, 1.50000, 0.15000,
-1.5000, 0.84000, 0.15000,
-1.5000, 0.00000, 0.15000,
-0.8400, 1.50000, 0.07500,
-1.5000, 0.84000, 0.07500,
-1.5000, 0.00000, 0.07500,     /* 288 */

-0.7980, 1.42500, 0.00000,
-1.4250, .798000, .000000,
-1.4250, .000000, .000000,
-1.5000, -.84000, .150000,
-.84000, -1.5000, .150000,
.000000, -1.5000, .150000,
-1.5000, -.84000, .075000,
-.84000, -1.5000, .075000,
.000000, -1.5000, .075000,
-1.4250, -.79800, .000000,
-.79800, -1.4250, .000000,
.000000, -1.4250, .000000,
.840000, -1.5000, .150000,
1.50000, -.84000, .150000,
.840000, -1.5000, .075000,
1.50000, -.84000, .075000,     /* 304 */

.798000, -1.4250, .000000,
1.42500, -.79800, .000000      /* 306 */
```

REDECOUVERTE ASM

```
;*****************************************************************************
;*      Programme CAPTAUX.A86
;*
;*
;*
;*        Maquette de lecture d'un port AUX et d'enregistrement des données
;*        reçues dans un fichier appelé CAPTAUX.n où n est le n° d'AUX
;*
;*
;* ---------------------------------
;*        NIVEAUX
;*
;*
;*        CRE      12-MAR-90      PhD      v. 0
;* ---------------------------------
;*        INVENTAIRE DES VARIABLES
;*
;*
;*        ADR_MSG        2      adresse du message de console
;*        CARACTERE      1      caractère reçu sur le port AUX
;*        FCB           36      "File Control Block", descripteur du fichier
;*        NUM_AUX        1      numéro du port AUX connu du système
;*        CANAL          1      traduction ASCII du no de port AUX
;*        TAMPON       128      contenu de l'enregistrement a faire
;*        TAILLE         1      nombre de caracteres deposes dans le TAMPON
;* ---------------------------------
;*        INVENTAIRE DES INDICATEURS      (VRAI=1, FAUX=0)
;*
;*
;*        ARRET          pour sortir
;*        AUX_OK         etat du port AUXilliaire
;*        ERREUR         gere par les fonctions critiques
;*        VIDE           etat du TAMPON
;* ---------------------------------
;*        INVENTAIRE DES ACTIONS
;*
;*
;*        QUEL_AUX      renvoi no de port connu du système dans NUM_AUX
;*
;*        PREND_AUX     attache le port AUX connu à la procèdure
;*                      met à jour l'indicateur ERREUR et la variable ADR_MSG
;*
;*        TST_AUX       renvoi AUX_OK vrai si un caractère est à lire
;*
;*        LIT_AUX       capte le caractère disponible dans CARACTERE
;*
;*        ECHO_CON      fait echo du caractere recu a la console active
;*
;*        ECHO_MSG      envoi a l'ecran le message dont l'adresse est contenue
;*                      dans la variable ADR_MSG
;*
;*        MAJ_TAMPON    charge le caractere recu dans le tampon a enregistrer
;*
;*        ECRIT_TAMPON  enregistre le tampon dans le fichier
;*                      met a jour l'indicateur ERREUR
;*                      met a jour la variable ADR_MSG
;*
;*        INI_TAMPON    initialise la variable TAMPON avec des CTRL-Z
;*                      rend l'indicateur VIDE vrai
;*
;*        TST_TAMPON    met l'indicateur VIDE faux si le tampon doit etre
;*                      enregistre avant d'etre mis a jour
;*
;*        TST_CON       met l'indicateur ARRET vrai si le caractere touche au
;*                      clavier de la console active est CTRL-C
;*
;*        OUVRIR        ouverture (ou creation) du fichier recepteur decrit
;*                       dans la variable FCB
;*                       met a jour l'indicateur ERREUR
;*
```

```
;*              met a jour la variable ADR_MSG
;*
;*    CLORE     ferme le fichier decrit dans FCB
;*              met a jour l'indicateur ERREUR
;*              met a jour la variable ADR_MSG
;*
;*    SORTIE    envoie le message de fin d'execution
;*              arrete la procedure
;
;*    ALGORITHME
;*
;*
;*    1.  recuperer no d'AUX connu du systeme
;*        envoyer message d'accueil
;*        attacher l'AUX connu a la procedure
;*        s'il n'est pas disponible, commenter et sortir
;*
;*    2.  ouverture (creation) du fichier recepteur ;
;*        en cas d'erreur critique, commenter et sortir ;
;*        initialiser le tampon d'enregistrement
;*
;*    3.  tant que le clavier n'a pas emis ESC, lire le port AUX,
;*        faire echo du caractere lu vers la console active, puis
;*        mettre a jour le tampon d'enregistrement apres
;*        avoir teste la place disponible. Si le tampon est plein,
;*        l'enregistrer d'abord dans le fichier recepteur.
;*
;*    4.  fin de procedure : enregistrer le tampon en cours, clore le
;*        fichier recepteur, envoyer le message de fin d'execution,
;*        liberer le port auxiliaire, et stopper la procedure.
;*
;*    FONCTIONS SYSTEME UTILISEES

;    1. relatives au port auxilliaire

a_get           equ     169     ; récupère dans AL le no d'AUX connu du système
a_cattach       equ     167     ; attache conditionnellement l'AUX connu
a_statin        equ     7       ; teste l'AUX pour un caractere pret a etre lu
a_read          equ     3       ; lecture du caractere pret au port AUX
a_detach        equ     166     ; rend l'AUX disponible a d'autres activites


;    2. relatives a la console active

c_writestr      equ     9       ; envoi d'un string a l'ecran (fin de string=$)
c_write         equ     2       ; envoi d'un caractere a l'ecran actif
c_stat          equ     11      ; teste le clavier pour un caractere pret
c_rowio         equ     6       ; lit le caractere disponible au clavier
c_rowio_mode    equ     0ffh    ; selon le mode sans echo


;    3. relatives au(x) fichier(s)

f_err           equ     45      ; reaction du systeme d'exploitation aux erreurs
f_err_mode      equ     0feh    ; en mode gere par la procedure
f_open          equ     15      ; ouverture du fichier decrit dans FCB
f_make          equ     22      ; creation d'un nouveau fichier decrit dans FCB
f_dmaoff        equ     26      ; regle l'adresse de DMA (ici a celle du TAMPON)
f_dmaseg        equ     51      ; regle le segment de DMA (ici celui du TAMPON)
f_write         equ     21      ; enregistre le contenu de DMA (ici le TAMPON)
f_close         equ     16      ; cloture du fichier decrit dans FCB


;    4. constantes utilisees

eof             equ     26      ; CTRL-Z, marqueur de fin de fichier
cr              equ     13      ; CTRL-M, retour charriot
lf              equ     10      ; CTRL-J, montee de ligne
tab             equ     9       ; CTRL-I, tabulateur
ctrlc           equ     3       ; CTRL-C, interruption du programme
ccpm            equ     224     ; appel du systeme d'exploitation pour execution

;=============================================================================
;    PROGRAMME
;=============================================================================

un:         call        quel_aux
            mov         adr_msg,offset accueil_msg
            mov         extens,al
            call        echo_msg

            ;           mettre comme extension du nom de fichier le numero de canal
            mov         bx,offset fcb
            mov         si,9    ; position du premier caractere de l'extension
            mov         al,canal
            mov         [bx+si],al

            call        prend_aux
            cmp         erreur,0
            jz          un_1

            call echo_msg
            jmpf sortie

un_1:       call        echo_msg

;-------

deux:       call        ouvrir
            cmp         erreur,0
            jz          deux_1

                        call echo_msg
                        jmpf sortie

deux_1:     call        ini_tampon
            mov         arret,0

;-------

trois:      call        tst_con
            cmp         arret,1         ; s'arreter ?
            jnz         trois_1         ; non

                        jmp quatre      ; sortie

trois_1:  call tst_aux
            cmp aux_ok,1                ; un caractere est pret ?
            jz trois_2                          ; oui

                        jmps trois

trois_2:call lit_aux
            call echo_con

            call tst_tampon
            cmp vide,1
            jz trois_3

                        call ecrit_tampon
                        call ini_tampon
                        cmp erreur,0
                        jz trois_3

                                    call echo_msg
                                    jmpf clore

trois_3:  call maj_tampon
            jmps trois              ;saut très rapide dans un espace de 128 bytes


quatre:     call ecrit_tampon
            cmp erreur,0
            jz clore

                                    call echo_msg

;=============================================================================
;    ACTIONS
;=============================================================================


clore:      mov                     cl,f_close
            mov                     dx,offset fcb
            call                    systeme

libere_aux:
            mov                     cl,a_detach
            call                    systeme

sortie:     mov                     adr_msg,offset sortie_msg
            call                    echo_msg

            mov                     al,0
            mov                     bl,0
            mov                     cl,0
            mov                     dl,0
            jmpf                    systeme

;-------

quel_aux:
            mov                     cl,a_get
            call                    systeme
            mov                     num_aux,al
            add                     al,48
            mov                     canal,al
            ret

;-------

prend_aux:
            mov                     erreur,0
            mov                     adr_msg,offset dispo
            mov                     cl,a_cattach
            call                    systeme
            cmp                     ax,0ffffh
            jnz                     prend_aux_1

            mov                     erreur,1
            mov                     adr_msg,offset non_dispo

prend_aux_1:
            ret

;-------

tst_aux:
            mov                     aux_ok,1
            mov                     cl,a_statin
            call                    systeme
            cmp                     al,0ffh
            jz                      tst_aux_1

            mov                     aux_ok,0

tst_aux_1:
            ret
```

```
;--------

lit_aux:
     mov          cl,a_read
     call         systeme
     mov          caractere,al
     ret

;--------

echo_con:
     mov          cl,c_write
     mov          dl,caractere
     call         systeme
     ret

;--------

echo_msg:
     mov          cl,c_writestr
     mov          dx,adr_msg
     call         systeme
     ret

;--------

maj_tampon:
     mov          si,taille
     mov          bx,offset tampon
     mov          al,caractere
     mov          [bx+si],al
     inc          taille
     ret

;--------

ecrit_tampon:
     mov          cl,f_dmaoff
     mov          dx,offset tampon
     call         systeme

     mov          cl,f_dmaseg
     mov          dx,seg tampon
     call         systeme

     mov          cl,f_write
     mov          dx,offset fcb
     call         systeme
     mov          erreur,0
     cmp          al,0                    ; erreur a l'enregistrement ?
     jz           retour_ecr              ; non

                  cmp       al,0ffh       ; erreur physique ?
                  jnz       er_file       ; non, alors erreur fichier
                            call     err_dsk     ; autres erreurs disque

retour_ecr:
     ret

;--------

er_file:
     call         erf1
     jmp          retour_ecr


erf1: cmp         al,1                    ; pas de place dans le repertoire
      jnz         erf2

                  mov     adr_msg,offset erf1_msg
                  mov     erreur,1
                  ret

erf2: cmp         al,2                    ; disque plein
      jnz         erf8

                  mov       adr_msg,offset erf2_msg
                  mov       erreur,1
                  ret

erf8: cmp         al,8                    ; enregistrement verrouille par autre activite
      jnz         erf9

                  mov     adr_msg,offset erf8_msg
                  mov     erreur,1
                  ret

erf9: cmp         al,9                    ; le ? est interdit dans nom fichier
      jnz         erf10

                  mov     adr_msg,offset erf9_msg
                  mov     erreur,1
                  ret

erf10: cmp        al,10                   ; erreur CRC sur le FCB
       jnz        erf11

                  mov     adr_msg,offset erf10_msg
                  mov     erreur,1
                  ret

erf11: cmp        al,11              ; erreur verification fichier non verrouille
       jnz        erf_inc
       mov        adr_msg,offset erf11_msg
                  mov     erreur,1
                  ret


erf_inc:mov       adr_msg,offset erf_inc_msg     ; erreur de nature inconnue
          mov     erreur,1
          ret

;--------

err_dsk:  cmp     ah,1                    ; erreur disque I/O
          jnz     erec2

                  mov     adr_msg,offset erec1_msg
                  mov     erreur,1
                  ret

erec2:    cmp     ah,2                    ; protection disque
          jnz     erec3

                  mov     adr_msg,offset erec2_msg
                  mov     erreur,1
                  ret

erec3:    cmp     ah,3                    ; fichier protege
          jnz     erec4

                  mov     adr_msg,offset erec3_msg
                  mov     erreur,1
                  ret

erec4:    cmp     ah,4                    ; erreur selection disque
          jnz     erec_inc

                  mov     adr_msg,offset erec4_msg
                  mov     erreur,1
                  ret

erec_inc:
          mov     adr_msg,offset erec_inc_msg
          mov     erreur,1
          ret

;--------

ini_tampon:
          mov     si,0
          mov     bx,offset tampon

ini_tampon_1:
          mov     byte [bx+si],26
          inc     si
          cmp     si,128
          jnz     ini_tampon_1
          mov     vide,1
          mov     taille,0
          ret

;--------

tst_tampon:
          mov     vide,1
          mov     si,128
          cmp     taille,si
          jnz     tst_tampon_1

          mov     vide,0

tst_tampon_1:
          ret

;--------

tst_con:
          mov     arret,0
          mov     cl,c_stat
          call    systeme
          cmp     al,1
          jnz     tst_con_1

                  mov       cl,c_rowio
                  mov       dl,c_rowio_mode
                  call      systeme
                  cmp       al,27
                  jnz       tst_con_1

                            mov     arret,1

tst_con_1:
          ret

;--------

ouvrir:
          mov     cl,f_err          ; declare le mode gestion des erreurs
          mov     dl,f_err_mode
          call    systeme

          mov     cl,f_open         ; ouverture
          mov     dx,offset fcb
          call    systeme
          cmp     al,0              ; le fichier existe-t-il deja ?
          jnz     cre_fic           ; non, alors le creer

                  mov     si,32
```

```
        mov     bx,offset fcb
        mov     byte [bx+si],0      ; forcer ecriture a debut du fichier
        ret

cre_fic:
        mov     erreur,0
        mov     cl,f_make
        mov     dx,offset fcb
        call    systeme
        cmp     al,0                ; creation ok ?
        jz      cre_fic_1           ; oui

                cmp     ah,1        ; erreur permanente disque
                jnz     er_crefic_2

                        mov     erreur,1
                        mov     adr_msg,offset er_cref1_msg

cre_fic_1:      ret

er_crefic_2:    cmp     ah,2    ; disque lisible seulement
                jnz     er_crefic_4

                        mov     erreur,1
                        mov     adr_msg,offset er_cref4_msg
                        ret

er_crefic_4:    cmp     ah,4        ; selection de disque erronnee
                jnz     er_crefic_8

                        mov     erreur,1
                        mov     adr_msg,offset er_cref4_msg
                        ret

er_crefic_8:    cmp     ah,8    ; fichier existant deja
                jnz     er_crefic_9

                        mov     erreur,1
                        mov     adr_msg,offset er_cref8_msg
                        ret

er_crefic_9:    cmp     ah,9        ; caractere illicite dans le nom de fichier
                jnz     er_crefic_10

                        mov     erreur,1
                        mov     adr_msg,offset er_cref9_msg
                        ret

er_crefic_10:   cmp     ah,10   ; trop de fichiers ouverts
                jnz     er_crefic_11

                        mov     erreur,1
                        mov     adr_msg,offset er_cref10_msg
                        ret

er_crefic_11:   cmp     ah,11   ; table de verrouillage pleine
                jnz     er_cref_inc

                        mov     erreur,1
                        mov     adr_msg,offset er_cref11_msg
                        ret

er_cref_inc:    mov     erreur,1    ; erreur inconnue en mode creation
                mov     adr_msg,offset er_crefinc_msg
                ret

;_______

systeme:
        int 224
        ret

;=============================================================================
;       VARIABLES
;=============================================================================

        dseg
num_aux     rb 1
adr_msg dw 0

accueil_msg         db 01bh,'z',01bh,'E',01bh,'H',0dh
                    db 09h,09h,'Lecture de donnees sur le port AUX actif',0dh,0ah
                    db 09h,09h,'-----------------------',0dh,0ah
                    db 09h,09h,09h,'      PhD, le 12.03.90',0dh,0ah,0ah,0ah
                    db 0dh,0ah,'Le port AUX a lire est le `
canal               db `0'
                    db 0dh,0ah,'Les donnees seront enregistrees dans le fichier CAPTURE.'
extens              db `0'
                    db 0dh,0ah,'On sort avec cloture par ESC, ou sans avec CTRL-C'
                    db 0dh,0ah,0ah,'$'

dispo               db 13,10,7,9,'Capture de donnees en cours.',13,10,'$'
non_dispo           db 0dh,0ah,'Le port n''est pas disponible$'
existe              db 13,10,7,9,'Fichier contenant deja des donnees$'

erf_inc_msg         db 13,10,7,9,'Erreur inconnue en ecriture dans le fichier$'
erf1_msg            db 13,10,7,9,'Pas de place dans le repertoire$'
erf2_msg            db 13,10,7,9,'Pas de place sur le disque$'
erf8_msg            db 13,10,7,9,'Enregistrement verrouille par une autre activite$'
erf9_msg            db 13,10,7,9,'Il y a un caractere interdit dans le nom du fichier$'
erf10_msg           db 13,10,7,9,'Erreur de CRC sur la description du fichier$'
erf11_msg           db 13,10,7,9,'Erreur verification fichier non verrouille$'

erec1_msg           db 13,10,7,9,'Erreur permanente I/O disque$'
erec2_msg           db 13,10,7,9,'Disque protege en ecriture$'
erec3_msg           db 13,10,7,9,'Fichier protege en ecriture$'
erec4_msg           db 13,10,7,9,'Erreur dans la selection du disque$'
erec_inc_msg        db 13,10,7,9,'Erreur de type inconnu en mode ecriture$'
er_cref1_msg        db 13,10,7,9,'Erreur disque irrecuperable$'
er_cref2_msg        db 13,10,7,9,'Disque accessible en lecture seulement$'
er_cref4_msg        db 13,10,7,9,'Erreur de selection du disque$'
er_cref8_msg        db 13,10,7,9,'Creation d''un fichier existant$'
er_cref9_msg        db 13,10,7,9,'Caractere illicite dans le nom du fichier$'
er_cref10_msg       db 13,10,7,9,'Trop de fichiers ouverts$'
er_cref11_msg       db 13,10,7,9,'Table de verrouillage pleine$'
er_crefinc_msg      db 13,10,7,9,'Erreur de type inconnu en creation de fichier$'

arret       rb 1                ; flag fin de procedure
aux_ok      rb 1                ; flag test auxilliaire
vide        rb 1                ; flag tampon vide
erreur      rb 1                ; flag erreur
caractere   rb 1

sortie_msg          db 0dh,0ah,0ah,09h,'*** FIN DE PROCEDURE ***',0dh,0ah,0ah,'$'

msgtmp      db 27,89,51,32
tampon      rb 128
taille      dw 0

;           disk    fichier
fcb         db 0,     'CAPTURE 0  '

;           ex cs rs rc d0 ---------- d15  cr r0 r1 r2
        db 0, 0, 0, 0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,    0, 0, 0, 0

end
```

UN NOYAU MULTI-TACHES EN C++

[LISTING UN]

```
/*******************************************/
/*                  TASK.HPP               */
/*                  Tom Green              */
/*******************************************/
/* Ce fichier contient les classes dont use le noyau multi-taches */
/* L'inclure dans votre code source et le lier a TASK.CPP et TIMER.ASM */

/* Ce qui suit sert a l'initialisation d'une tache */
/* c'est un pointeur de fonction */
typedef void (*func_ptr)(void);

/* Ceci sert au gestionnaire d'interruptions a appeler l'ancien */
/* c'est un pointeur a distance de fonction */
typedef void (far *far_func_ptr)(void);

/* Voici comment paraitront les registres sur la pile d'une tache */
/* apres leur sauvegarde pour changement de tache. Les registre SP et SS */
/* pointeront la-dessus quand une tache sera lancee par le gestionnaire */
/* d'interruption ou par save_image */

typedef struct task_image{
        unsigned int bp;
        unsigned int di;
        unsigned int si;
        unsigned int ds;
        unsigned int es;
        unsigned int dx;
        unsigned int cx;
        unsigned int bx;
        unsigned int ax;
        unsigned int ip;
        unsigned int cs;
        unsigned int flags;
}task_image;

/* Un objet "Tache". Il contient les informations utiles a l'objet */
/* "task_control", pour pratiquer la commutation de taches ainsi */
/* qu'un pointeur sur l'aire de travail de la tache (pile) */

class task{
        private:
                friend class task_control;  // acces utile a l'objet task_control
                friend class signal;        // acces de "signal" a "next_task"
                task_image far *stack_ptr;  // pointeur sur l'"image" de la pile de la tache
                unsigned char task_state;   // indicateur d'etat de tache
                unsigned char *workspace;   // adresse de la pile reservee de la tache
                task *next_task;            // pointeur sur la prochaine tache dans la file
        public:
                task(func_ptr func,unsigned int workspace_size); //
constructor
                ~task();
```

```
                                                // destructor
};


/* File d'attente d'une tache */
/* On la nomme "signal" a fin de definir un signal de communication inter-taches */

class signal{
        private:
                friend class task_control;   // task_control needs access
                task *head;
                task *tail;
                task *get_task_q(void);      // get next task off of queue
                void put_task_q(task *tptr); // append task to queue
        public:
                signal(void){head=tail=0;};  // constructor
};



/* Objet "task_control" (gestionnaire de taches) */
/* Procedures et methodes d'interfacage et de gestion des taches */
/* Cet objet initialise ou retablit les vecteurs d'interruption, */
/* prend soin des tops d'horloge et commute les objets-taches a  */
/* executer */

class task_control{
        private:
                signal ready_q;                 // file des taches pretes
                task *current_task;             // tache en action
                task_image far *old_stack_ptr; // revient a cette pile en fin
                unsigned int task_running;      // activation de l'indicateur de commutation de tache
                unsigned long timer_ticks;      // 18.2 tops/seconde
                unsigned int task_lock;         // verrouille la commutation
                task_image far *task_switch(task_image far *stk_ptr,

                unsigned int flag,

                signal *sig);
        public:
                task_control(void);                     // constructeur
                void add_new_task(task *new_task);      // ajoute un tache a la file des taches pretes
                void start_tasks(void);                 // demarre la commutation de taches sur la file
                void stop_tasks(void){task_running=0;};
                unsigned                                                long
get_timer_ticks(void){return(timer_ticks);};
                void lock(void){task_lock=1;};  // la tache en action ne peut etre commutee
                void unlock(void){task_lock=0;};// autorise la commutation de taches
                void send(signal *sig);         // transfere une tache de la file "signal" a la file "ready"
                void wait(signal *sig);         // depose une tache dans la file "signal"
                void block(void);               // une tache en autorise une autre a fonctionner
};



[LISTING DEUX]

/*******************************************/
/*                  TASK.CPP               */
/*                  by Tom Green           */

/*******************************************/

/* Ce fichier installe les methodes dont se servent le gestionnaire */
/* de taches et les objets-taches */

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <int.h>
#include "task.hpp"

/* Etats d'une tache */
#define   TASK_INACTIVE        0
#define   TASK_ACTIVE          1
#define TASK_READY   2
#define   TASK_WAITING         3
#define TASK_ERROR   0xff

/* indicateurs d'interfacage des procedures */
#define   TASK_TIMER_INTR      0
#define   TASK_SEND            1
#define   TASK_WAIT            2
#define TASK_BLOCK             3

/* systeme d'interruption du compte-temps ou "tops d'horloge" */
#define TIMER_INT              8

/* procedures dont nous avons besoin et provenant de TIMER.ASM */
unsigned int getCS(void);
extern void timer_handler(void);
extern void save_image(unsigned int flag,signal *sig);

/* "global" pour que le gestionnaire d'horloge appelle l'ancienne */
/* procedure de gestion d'interruption d'horloge */
far_func_ptr old_timer_handler;

/* Ce qui suit est franchement horrible. */
/* Quand le constructeur d'un objet "task_control" est appele, on sauvegarde */
/* le "this pointer" pour que la procedure de commutation de taches appelle */
/* notre objet de gestion de taches "task_switch". */
/* Cela signifie qu'on ne peut avoir qu'un seul objet "task_control". Desole ! */
task_control *gl_tptr;

/* Constructeur d'une tache. L'espace de travail en sera l'espace de la pile */
/* de la tache. Quand survient l'interruption du compte-temps, l'"image" */
/* est empilee pour usage ulterieur, et le pointeur *stack_ptr de task_image */
/* designera cette image */

task::task(func_ptr func,unsigned int workspace_size)
{
        task_image *ptr;

        /* get stack or "workspace" for task */
        if((workspace=(unsigned char *)malloc(workspace_size))==NULL){
                task_state=TASK_ERROR;          // do not let this one run
                return;
        }

        /* Maintenant, on dessine l'image de depart des registres de la tache */
        /* ptr vise l'image des registres pour demarrer la tache */
        ptr=(task_image *)(workspace+workspace_size-sizeof(task_image));

        /* puis on sauve le pointeur visant l'image des registres */
        stack_ptr=MK_FP(getDS(),(unsigned int)ptr);

        ptr->ip=(unsigned int)func; // decalage du pointeur de code de la tache
        ptr->cs=getCS();            // segment du pointeur de tache, "bug" du compilateur
        ptr->ds=getDS();
        ptr->flags=0x200;       // indicateurs, interruptions HAUTS
        task_state=TASK_INACTIVE;   // la tache est inactive
        next_task=0;

/* destructeur d'un objet-tache */

task::~task(void)
{
        free(workspace);
}

/* capture la prochaine tache inactive dans la file des taches */

task *signal::get_task_q(void)
{
        task *temp;

        temp=head;
        if(head)
                head=head->next_task;
        return(temp);
}

/* ajoute une tache en bout de file d'attente */

void signal::put_task_q(task *tptr)
{
        if(head)
                tail->next_task=tptr;
        else
                head=tptr;
        tail=tptr;
        tptr->next_task=0;
}

/* constructeur de "task_control" */
/* initialise les moyens locaux de controle d'une tache */

task_control::task_control(void)
{
        gl_tptr=this;
        task_running=0;
        current_task=0;
        timer_ticks=0L;
        task_lock=0;
}

/* ajoute une tache a la file "ready_q" */
/* cette procedure est appelee apres creation d'un objet-tache */

void task_control::add_new_task(task *new_task)
{
        if(new_task->task_state!=TASK_ERROR){
                new_task->task_state=TASK_READY;
                ready_q.put_task_q(new_task);
        }
}

/* appel pour lancer les taches apres qu'on en ait creees */
/* et qu'on les ait ajouter a la file "ready_q" */

void task_control::start_tasks(void)
{
        unsigned int offset,segment;

        task_running=1;

        /* recuperer l'adresse de l'ancien gestionnaire d'interruption d'horloge */
        int_getvector(TIMER_INT,&offset,&segment);
        old_timer_handler=(far_func_ptr)(MK_FP(segment,offset));

        /* mise en place de notre nouveau gestionnaire d'horloge */
        int_setvector(TIMER_INT,(unsigned int)timer_handler,getCS());

        /* maintenant, on peut lancer les taches */
        while(task_running)
                ; // ne fait rien, truc permettant d'attendre le lancement des taches
        /* vient ici quand l'excution multi-taches est arretee */
```

```
}

/* Recupere la prochaine tache inactive dans la file "signal" et la depose */
/* dans la file "ready_q". Cela suspend l'action de la tache appelante, */
/* laquelle est aussi deposee dans la fille des taches pretes */

void task_control::send(signal *sig)
{
        save_image(TASK_SEND,sig);
}

/* Depose la tache appelante dans la file des taches attendant un signal */

void task_control::wait(signal *sig)
{
        save_image(TASK_WAIT,sig);
}

/* Ceci provoque la mise de la tache en cours dans la file des taches pretes */
/* et la commutation sur la tache suivante dans "ready_q" */

void task_control::block(void)
{
        save_image(TASK_BLOCK,(signal *)0);
}

/* Cette procedure est appelee pour commuter des taches. On lui passe */
/* un pointeur "far" visant la pile en cours ou "image" de la tache. */
/* On passe aussi un indicateur (decrit plus loin) et, si necessaire */
/* un pointeur de signal. if needed. */
/* Elle renvoie un pointeur sur l'"image" de la tache suivante. */

task_image far *task_control::task_switch(task_image far *stk_ptr,

                                          signal *sig)
{
        task_image far *temp;
        task *tptr;

        if(flag==TASK_TIMER_INTR)  // increment clock if it is a timer interrupt
                timer_ticks++;

        /* Ceci sauve un pointeur de pile des le lancement du multi-taches */
        /* Ca nous permet de revenir au point d'ou "start_tasks" avait ete appelee */
        if(!current_task){  // no current task so save state for restoring
                old_stack_ptr=stk_ptr;          // save stack pointer
                current_task=ready_q.get_task_q(); // set up a current task
        }

        /* Une tache est active, commuton si c'est possible */
        if(current_task->task_state==TASK_ACTIVE){
                current_task->stack_ptr=stk_ptr;  // sauve le pointeur de pile
                current_task->task_state=TASK_READY; // la tache est prete a fonctionner
                /* ne pas autoriser la commutation si des taches sont */
                /* verrouillees et qu'il y a interruption d'horloge */
                if(!task_lock || flag!=TASK_TIMER_INTR){
                        /* voir qui a bien pu provoquer l'appel de
"task_switch" */
                        switch(flag){
                                case TASK_WAIT:
                        current_task->task_state=TASK_WAITING;
                        sig->put_task_q(current_task);
                                        break;
                                case TASK_SEND:
                        if((tptr=sig->get_task_q())!=0)

ready_q.put_task_q(tptr);
                                        // et on se retrouve ici
                                case TASK_BLOCK:
                                case TASK_TIMER_INTR:
                                current_task->task_state=TASK_READY;
                /* depose la tache precedente dans la file "ready_q" */

ready_q.put_task_q(current_task);
                                        break;
                        }
                        /* recupere la tache suivante a traiter */
                        current_task=ready_q.get_task_q();
                }
        }

        /* si nous sommes toujours en multi-taches, prend la tache prete a l'action */
        if(task_running){
                current_task->task_state=TASK_ACTIVE;
                temp=current_task->stack_ptr;  // recupere le pointeur de pile de la tache
        }
        /* le multi-tachaes ayant ete stoppe, se preparer a revenir au
point de depart */
        else{                           // quelqu'un a appele "stop_tasks"
                int_setvector(TIMER_INT,FP_OFF(old_timer_handler),

FP_SEG(old_timer_handler));
                temp=old_stack_ptr;  // restitue la pile originelle
        }
        /* renvoie le pointeur "far" d'image de pile pour changer de tache */
        return(temp);
}
```

[LISTING TROIS]

```
;*****************************************************************************
;       TIMER.ASM
;       by Tom Green
;       Gestionnaire d'interruptions d'horloge
;       Ce gestionnaire appele tout d'abord le gestionnaire originel puis
;       l'objet "task_control" de commutation de taches. Un pointeur sur l'"image"
;       de la pile de la nouvelle tache est renvoye par le commutateur de taches
;
;       "getCS" renvoie le segment du code actuel
;       "save_image" conserve l'"image" de la tache, comme si une interruption
;       etait survenue, puis appelle l'objet de commutation de taches "task_control".
;       Un pointeur visant l'"image" de la pile de la nouvelle tache est
;       renvoye par le commutateur de taches.
;*****************************************************************************

        .MODEL SMALL
        .8086

        .DATA

extrn _old_timer_handler:dword
extrn _gl_tptr:word
extrn __task_control_task_switch:near

        .CODE

;*****************************************************************************
;       l'entier non signe getCS(void) rend le segment du code en cours.
;       C'est pour pallier une erreur du compilateur. Quand une fonction mise
;       a disposition d'un fonction "far" et qu'on veuille en recueillir le segment,
;       DS est renvoye en place de CS.
;*****************************************************************************
_getCS  proc    near
public  _getCS
                mov             ax,cs
                ret
_getCS  endp

;*****************************************************************************
;       timer_handler - Ceci remplace l'interrupteur top d'horloge de MS-DOS (8H).
;       Cette procedure sauve tout ce qu'il y a sur la pile, appelle le gestionnaire
;       original d'interruptions d'horloge, puis appelle l'objet commutateur
;       de taches "task_control".
;*****************************************************************************
_timer_handler  proc    near
public  _timer_handler
                push    ax              ;sauve ce qu'il y a
                push    bx
                push    cx
                push    dx
                push    es
                push    ds
                push    si
                push    di
                push    bp
                mov     bp,dgroup
                mov     ds,bp           ;recupere notre segment de donnees
                pushf
                call    dword ptr dgroup:_old_timer_handler  ;call original handler
                sti
                xor     dx,dx
                mov     ax,ss
                mov     bx,sp
                push    dx              ;pose 0 pour les 2 derniers parametres
                push    dx              ;representant l'interruption d'horloge
                push    ax
                push    bx
                mov     dx,_gl_tptr     ;pointeur cache pour les objets C++
                push    dx
;la pile est maintenant preparee pour task_switch appelle l'objet "task_control"
                cli                     ;inhibe les interruptions pour commuter les taches
                call    __task_control_task_switch
;il n'est pas utile de nettoyer la pile puisqu'elle va changer
                sti
                mov             ss,dx   ;le nouveau segment de pile se trouvait dans dx
                mov             sp,ax   ;et le nouveau pointeur de pile, dans ax
                pop             bp      ;retablit les registres
                pop             di
                pop             si
                pop             ds
                pop             es
                pop             dx
                pop             cx
                pop             bx
                pop             ax
                iret
_timer_handler  endp

;*****************************************************************************
;       void save_image(unsigned int flag,signal *sig) - Send, wait, block
;       etc, appels passant tous par ici pour sauver l'"image" de la tache.
;       Ce code simule ce qui arrive, quand il y a interruption, en sauvant
;       l'image de la tache sur la pile. L'indicateur passe est mis dans
;       l'objet de commutation de tache "task_control", ainsi sait-il s'il
;       a ete appele par le gestionnaire d'interruptions d'horloge, send,
;       wait ou block, etc. Le second parametre est un pointeur de signal,
;       dont se servent send et wait, et qui est passe via le commutateur
;       de taches.
;*****************************************************************************
_save_image             proc    near
public  _save_image
;Depuis un appel C, on risque de perdre quelques registres (ax bx cx dx),
;donc on va amenager la pile comme si une interruption etait intervenue.
;Rendre les parametres sur la pile, parce que la procedure appelante s'y referera au
;retour. bx et cx contiendront les parametres qui avaient ete passes.
                pop             ax ;prend sur la pile le decalage de l'adresse de retour
                pop             bx  ;preleve le premier parametre sur la pile
                pop             cx      ;y preleve le second
                push    cx              ;et les y redepose
                push    bx
```

```
                    pushf                           ;sauve les indicateurs pour "iret"
                    mov                 dx,cs       ;prend le segment du code
                    push        dx                  ;le sauve pour l'adresse de retour
                    push        ax                  ;depose le decalage prealablement sauve de l'adresse de retour
                    push        ax                  ;sauve tout le reste
                    push        bx
                    push        cx
                    push        dx
                    push        es
                    push        ds
                    push        si
                    push        di
                    push        bp
                    sti
                    mov                 ax,sp       ;parametre pointeur de pile
                    push        cx                  ;second parametre passe
                    push        bx                  ;premier parametre passe
                    mov                 bx,ss
                    push        bx                  ;pointeur "far" de pile, parametre passe
                    push        ax
                    mov                 ax,_gl_tptr ;pode le pointeur cache pour l'objet C++
                    push        ax
;on prepare maintenant la pile pour appeler l'objet controleur de tache "task_switch"
                    cli                             ;inhibe les interruptions pour commuter les taches
                    call        __task_control_task_switch
;pas la peine de nettoyer la pile parce qu'elle va changer.
                    sti
                    mov                 ss,dx       ;le nouveau segment de pile se trouvait dans dx
                    mov                 sp,ax       ;le nouveau pointeur de pile se trouvait dans ax
                    pop                 bp          ;retablit les registres
                    pop                 di
                    pop                 si
                    pop                 ds
                    pop                 es
                    pop                 dx
                    pop                 cx
                    pop                 bx
                    pop                 ax
                    iret
_save_image                 endp

                    end
```

LES BIBLIOTHEQUES DE LIAISONS DYNAMIQUES DDL SOUS WINDOWS

DYNAMIC LINK LIBRARIES UNDER MICROSOFT WINDOWS
par Margaret Johnson et Mark Solinski

[LISTING UN]

```
/*************************************************************************
 * MODULE: HELPLIB.C
 * COMMENTS: contient les fonctions Screen et  Topics (see HELPDLG.C)
 * UPDATE:   négliger les variables lpfnScreenDlgProc et lpfnTopicsDlgProc
 *           pendant les appels à MakeProcInstance() et
 *           FreeProcInstance() dans Screen() et dans Topics(). On n'est pas
 *           censé appeler ces functions dans une DLL.
 *************************************************************************/

#include  <windows.h>            /* utilisé par tous les modules ecrits pour

Windows */
#include  "helplib.h"          /* fichier d'inclusion de bibliothèques */
#define   HELPLIB
#include  "prothelp.h"          /* prototypes de fonction */
/*************************************************************************
 * variables externes

*************************************************************************/
extern HANDLE hInst;   /* réglé par la fonction d'initialisation dans
libinitc.c */
/*************************************************************************
 * variables globales

*************************************************************************/
LPSCREEN    lpsc;
BOOL        TOPICS;
BOOL        SCR;
/*************************************************************************
 * FONCTION: Screen
 * PROPOS :  Affiche le texte d'assistance relatif à un sujet
 *************************************************************************/
BOOL FAR PASCAL Screen( LPSCREEN sc )
{MSG      msg;
 HWND     hWnd;
 LockData(0);
 lpsc = sc;

 if (!(hWnd = CreateDialog(hInst,"HELP_BOX",GetActiveWindow(),
                           ScreenDlgProc)))
    return FALSE;
 while (GetMessage (&msg,NULL,0,0))
    {if (!IsDialogMessage(hWnd,&msg) )
       {TranslateMessage(&msg);
        DispatchMessage(&msg);
       }
    }

 if (TOPICS)
    {Topics(lpsc);
    }

 UnlockData(0);
 return TRUE;
}
/*************************************************************************
 * FONCTION: Topics
 * PROPOS :  présenter le sommaire des sujets d'assistance disponibles.
 *************************************************************************/
VOID FAR PASCAL Topics( LPSCREEN sc )

{LockData(0);
 lpsc = sc;

 DialogBox(hInst,"TOPICS_BOX",GetActiveWindow(),TopicsDlgProc);

 if (SCR)
    {Screen(lpsc);
    }

 UnlockData(0);
 return;
}
```

[LISTING DEUX]

```
/*************************************************************************
 * FICHIER: helplib.rc
 * PROPOS : ressource pour la DLL "helplib"
 *************************************************************************/
#include <style.h>
#include "helplib.h"
#define  TABGRP (WS_TABSTOP | WS_GROUP)

VSRUNTIME   TEXT     vsruntime.asc


CAVEATS     TEXT        caveats.asc

COOKBOOK    TEXT     cookbook.asc
REF         TEXT     ref.asc
A           TEXT     a.asc
B           TEXT     b.asc
C           TEXT     c.asc

STRINGTABLE
   BEGIN
       IDS_MEMERROR     "Out of Memory"
       VSRUNTIME,       "versus run time libraries"
       CAVEATS,         "caveats"
       COOKBOOK,        "cookbook"
       REF,             "references"
       A,               "a"
       B,               "b"
       C,               "c"
   END

rcinclude  HELP.dlg
rcinclude  TOPICS.dlg
```

[LISTING TROIS]

```
/*************************************************************************
 * FICHIER: helplib.h
 * PROPOS : fichier d'inclusion pour la DLL "helplib"
 *************************************************************************/
typedef struct {
     WORD   wScreen;
     WORD   wStart;
     WORD   wEnd;
    }SCREEN, FAR * LPSCREEN, NEAR *NPSCREEN;
#define ID_SCREEN_HELP          100
#define ID_LB_TOPICS            101
#define ID_NEXT_HELP            102
#define ID_PREVIOUS_HELP        103
#define ID_TOPICS_HELP          104
#define ID_SCROLL_HELP          105

#define VSRUNTIME               300
#define CAVEATS                 301
#define COOKBOOK                302
#define REF                     303
```

```
#define A                       500
#define B                       501
#define C                       502

#define IDS_MEMERROR            1000



[LISTING QUATRE]

/*************************************************************************
 * MODULE: HELPDLG.C
 * UPDATE: setScroll() n'était pas correctement listé.
 *************************************************************************/

#include <windows.h>          /* nécessaire à toutes les applications Windows
*/
#include "helplib.h"          /* fichier d'inclusion de bibliothèques */
#define  HELPLIB
#include "prothelp.h"         /* prototypes de fonctions */
#include "string.h"           /* strlen */

/*************************************************************************
 * permet les multiples écrans venant des applications ou instances
 *************************************************************************/
typedef struct screenStruct {
    WORD                wScreen;
    WORD                wStart;
    WORD                wEnd;
    int                 nPage;
    int                 nTopics;
    int                 nNumLines;
    HWND                hScroll;
    int                 nVscrollPos;
   }HELPSCREEN, *NPHELPSCREEN;
/*************************************************************************
 * variables locales
*************************************************************************/
#define   GWW_SCREENHANDLE  0
#define   MAXBUFLEN         80
#define   MAXLINES          250
#define   LOCAL             static
/************************************
 * variables de positionnement de la barre déroulante
************************************/
LOCAL int       nVscrollMax;
/************************************
 * tampon où conserver le texte d'assitance
************************************/
LOCAL char      szText[MAXLINES][MAXBUFLEN];
/*****************************************
 * informations d'écran
 *****************************************/
LOCAL NPHELPSCREEN sptr;
/*************************************************************************
 * prototypes de fonctions locales
*************************************************************************/
LOCAL VOID   NEAR getText          ( VOID );
LOCAL VOID   NEAR setScroll        ( VOID );
LOCAL VOID   NEAR setNewHelp       ( HWND );
LOCAL HANDLE NEAR setToScreen      ( HWND );
LOCAL BOOL   NEAR differentScreen  ( HWND );
LOCAL BOOL   NEAR initScreen       ( HWND );
LOCAL VOID   NEAR freeScreen       ( HANDLE );
/*************************************************************************
 * variables externes
*************************************************************************/
extern HANDLE     hInst; /* réglé par la fonction d'initialisation dans
libinitc.c */
extern LPSCREEN lpsc;  /* passée dedans par la fonction appelante */
extern BOOL     TOPICS;/* utilisé par ScreenDlgProc quand l'opérateur demande
les sujets */
extern BOOL     SCR;   /* utilisé par TopicsDlgProc quand l'opérateur demande
l'assistance */
/*************************************************************************
 * FONCTION: ScreenDlgProc
 * PROPOS :  Réagit aux boutons-poussoirs: Topics, Next, Previous, et
 *           Cancel dela boîte de dialogue Screen().

*************************************************************************/
BOOL  FAR  PASCAL  ScreenDlgProc(HWND hDlg, WORD wMessage, WORD wParam,
                                 LONG lParam)
 {int             i;
  static BOOL     bImoved=FALSE;
  HWND            hWndScreen;
  HANDLE          hScreen;
  switch(wMessage)
     {
      case WM_INITDIALOG:
        hScreen = setToScreen(hDlg);
        sptr->nTopics = sptr->wEnd - sptr->wStart + 1;
        sptr->hScroll = GetDlgItem(hDlg,ID_SCROLL_HELP);
        getText();
        LocalUnlock(hScreen);
        break;
      case WM_MOVE:
        bImoved=TRUE;
        break;
      case WM_VSCROLL:
        if (bImoved)
          {hScreen = setToScreen(hDlg);
           getText();
           setScroll();
           LocalUnlock(hScreen);
           bImoved=FALSE;
          }
        hWndScreen = GetWindow(hDlg,GW_CHILD);
        SendMessage(hWndScreen,WM_VSCROLL,wParam,lParam);
        break;
      case WM_COMMAND:
         hScreen = setToScreen(hDlg);
         hWndScreen = GetWindow(hDlg,GW_CHILD);
         switch(wParam)
           {
            case ID_PREVIOUS_HELP:
               sptr->wScreen = sptr->wStart + (sptr->wScreen-sptr->wStart+
                                (sptr->nTopics-1)) % sptr->nTopics;
               setNewHelp(hWndScreen);
               LocalUnlock(hScreen);
               break;
            case ID_NEXT_HELP:
               sptr->wScreen = sptr->wStart + (sptr->wScreen-sptr->wStart+1)
                                                %sptr->nTopics;
               setNewHelp(hWndScreen);
               LocalUnlock(hScreen);
               break;
            case ID_TOPICS_HELP:
               TOPICS = TRUE;
            case IDCANCEL:
               freeScreen(hScreen);
               DestroyWindow(hDlg);
               break;
            default:
               return FALSE;
           }


              break;

      case WM_ACTIVATE:  /* Quand le dialogue est actif, vérifie que
                          * le bon mode d'écran a été choisi.
                          */
         if (!wParam)
            break;
      case WM_PAINT: /* L'opérateur peut commuter les dialogues d'assistances
                      * entre n applications (ou n instances)
                      */
         hScreen = setToScreen(hDlg);
         if (differentScreen(hDlg))
           {getText();
           }
         if (WM_ACTIVATE == wMessage)
           {SetFocus(sptr->hScroll);
            setScroll();
           }
         LocalUnlock(hScreen);
         return FALSE;
      default:
         return FALSE;
     }
  return TRUE;
 }
/*************************************************************************
 * FONCTION: ScreenWndProc
 * PROPOS  : Répondre aux messages reçus par la fenêtre Screen().
 * Notes:    La gestion des messages WM_VSCROLL et WM_PAINT est dérivée de
 *           Programming Windows, par Charles Petzold, pp. 117-122.

*************************************************************************/
long FAR PASCAL  ScreenWndProc(HWND hWnd, WORD wMessage, WORD wParam,
                               LONG lParam)
{PAINTSTRUCT ps;
 LOCAL               int          xChar,yChar;
 LOCAL               int          yClient;
 LOCAL               TEXTMETRIC   tm;
 int                 i;


 int                 nVscrollInc;

 int                 nPaintBeg,nPaintEnd;
 HDC                 hDC;
 HANDLE              hScreen;
 switch(wMessage)
   {
    case WM_CREATE:
        if (!initScreen(hWnd))
            break;
        hDC = GetDC(hWnd);
        GetTextMetrics(hDC,&tm);
        ReleaseDC(hWnd,hDC);
        xChar = tm.tmAveCharWidth;
        yChar = tm.tmHeight + tm.tmExternalLeading;
        TOPICS = FALSE;
        break;
    case WM_SIZE:
        hScreen = GetWindowWord(hWnd,GWW_SCREENHANDLE);
        sptr = (NPHELPSCREEN)LocalLock(hScreen);
        yClient = HIWORD(lParam);
        sptr->nPage = yClient / yChar ;
        LocalUnlock(hScreen);
        break;
    case WM_SETFOCUS:
        hScreen = GetWindowWord(hWnd, GWW_SCREENHANDLE);
        sptr = (NPHELPSCREEN)LocalLock(hScreen);
        SetFocus(sptr->hScroll);
        LocalUnlock(hScreen);
        break;
    /* le message WM_VSCROLL est envoyé par ScreenDlgProc, qui préserve
```

```
   * toujours sptr
   */
  case WM_VSCROLL:
     switch (wParam)
        {
         case SB_TOP:

                        nVscrollInc = -sptr->nVscrollPos;

           break;
         case SB_BOTTOM:
           nVscrollInc = nVscrollMax - sptr->nVscrollPos;
           break;
         case SB_LINEUP:
           nVscrollInc = -1;
           break;
         case SB_LINEDOWN:
           nVscrollInc = 1;
           break;
         case SB_PAGEUP:
           nVscrollInc = min(-1, -sptr->nPage) ;
           break;
         case SB_PAGEDOWN:
           nVscrollInc = max(1, sptr->nPage) ;
           break;
         case SB_THUMBPOSITION:
           nVscrollInc = LOWORD(lParam) - sptr->nVscrollPos;
           break;
         default:
           nVscrollInc = 0;
        }
       if (nVscrollInc = max(-sptr->nVscrollPos,
         min(nVscrollInc, nVscrollMax - sptr->nVscrollPos)))
        {sptr->nVscrollPos += nVscrollInc;
         ScrollWindow(hWnd, 0, -yChar * nVscrollInc, NULL, NULL);
         UpdateWindow(hWnd);
         SetScrollPos(sptr->hScroll,SB_CTL,sptr->nVscrollPos,TRUE);
        }
       break;
   case WM_PAINT:

      hScreen =  GetWindowWord(hWnd, GWW_SCREENHANDLE);

      sptr = (NPHELPSCREEN)LocalLock(hScreen);
      BeginPaint( hWnd, &ps );
      nPaintBeg = max(0,sptr->nVscrollPos + ps.rcPaint.top/yChar - 1);
      nPaintEnd = min(sptr->nNumLines,
                  sptr->nVscrollPos + ps.rcPaint.bottom / yChar);
      for (i=nPaintBeg; i < nPaintEnd; ++i)
          {TextOut(ps.hdc,xChar,yChar * (1 - sptr->nVscrollPos+i),
               szText[i],strlen(szText[i]));
          }
      EndPaint(hWnd,&ps);
      LocalUnlock(hScreen);
      break;
   case WM_DESTROY:
      PostQuitMessage(0);
      break;
   default:
       return DefWindowProc( hWnd, wMessage, wParam, lParam );
  }
 return 0L;
}
/*************************************************************************
 * FONCTION: TopicsDlgProc
 * PROPOS  : Répondre à la boite de listage et aux messages des boutons-poussoirs
 *           de la boite de dialogue Topics().

*************************************************************************/
BOOL  FAR  PASCAL  TopicsDlgProc(HWND hDlg, WORD wMessage, WORD wParam,
                                 LONG lParam)
 {int i;
  LOCAL char szBuff[MAXBUFLEN];

  switch(wMessage)
     {
      case WM_INITDIALOG:
        SetSysModalWindow(hDlg);
        for (i=lpsc->wStart;i<=lpsc->wEnd;++i)
           {LoadString(hInst,i,szBuff,MAXBUFLEN);
            SendDlgItemMessage(hDlg,ID_LB_TOPICS,LB_ADDSTRING,0,
                                    (LONG)(LPSTR)szBuff);
           }
        SendDlgItemMessage(hDlg,ID_LB_TOPICS,LB_SETCURSEL,0,0L);
        SCR = FALSE;
        break;
      case WM_COMMAND:
         switch(wParam)
           {
            case ID_LB_TOPICS:
               if ((HIWORD(lParam)) != LBN_DBLCLK)
                  {break;
                  }
            case IDOK:
               lpsc->wScreen = (WORD)SendDlgItemMessage(hDlg,ID_LB_TOPICS,
                               LB_GETCURSEL,0,0L) + lpsc->wStart;
               SCR=TRUE;
            case IDCANCEL:
               EndDialog(hDlg,TRUE);

                        break;
            default:
               return FALSE;
            }
          break;
       default:
          return FALSE;
       }
  return TRUE;
 }
/*************************************************************************
 * FONCTION: getText
 * PROPOS  : Retrouve le texte d'assistance relatif à l'écran en cours dans
 *           le fichier-ressource de la bibliothèque Help

*************************************************************************/
LOCAL VOID NEAR getText( )
  {LPSTR   lpText,lpTextBeg;

     HANDLE  hRes;

    hRes = LoadResource(hInst,
           FindResource(hInst,MAKEINTRESOURCE(sptr->wScreen),"TEXT"));
    lpText = LockResource(hRes);
    sptr->nNumLines=0;
    lpTextBeg = lpText;
    while (*lpText != '\0' && *lpText != '\x1A' )
         {if (*lpText == '\r')
            {*lpText = '\0';
             lstrcpy(szText[sptr->nNumLines++],lpTextBeg);
             if (sptr->nNumLines >= MAXLINES)
                 break;
             *lpText='\r';
             lpText = AnsiNext(lpText);

                    if (*lpText = '\l')

                lpText = AnsiNext(lpText);
             lpTextBeg = lpText;
            }
          else
            lpText = AnsiNext(lpText);
         }
    *lpText = '\0';
    lstrcpy(szText[sptr->nNumLines++],lpTextBeg);
    GlobalUnlock(hRes);
    FreeResource(hRes);
   }
/*************************************************************************
 * FONCTION: setScroll
 * PROPOS  : Régler les limites de la barre déroulante et la position initiale.

*************************************************************************/
LOCAL VOID NEAR setScroll( )
{nVscrollMax = max(0, sptr->nNumLines + 2 - sptr->nPage);
 if (!nVscrollMax)
     {ShowScrollBar(sptr->hScroll,SB_CTL,FALSE);
     }
 else
     {SetScrollRange(sptr->hScroll, SB_CTL,0, nVscrollMax, FALSE);
      SetScrollPos  (sptr->hScroll, SB_CTL,sptr->nVscrollPos,TRUE);
      ShowScrollBar(sptr->hScroll,SB_CTL,TRUE);
     }
}



LOCAL VOID NEAR setScroll( )
  {nVscrollMax = max(0, sptr->nNumLines + 2 - sptr->nPage);
   SetScrollRange(sptr->hScroll, SB_CTL,0, nVscrollMax, FALSE);
   SetScrollPos  (sptr->hScroll, SB_CTL,sptr->nVscrollPos,TRUE);
  }
/*************************************************************************
 * FONCTION: setNewHelp
 * PROPOS  : Prend les infos propres à l'écran d'aide

*************************************************************************/
LOCAL VOID NEAR setNewHelp(HWND hWnd)
 {sptr->nVscrollPos = 0;
  getText();
  setScroll();
  InvalidateRect(hWnd,NULL,TRUE);
  SetFocus(sptr->hScroll);
 }
/*************************************************************************


  * FONCTION: setToScreen

  * PROPOS  : Verrouille le pointeur d'écran sur les bonnes infos d'écran

*************************************************************************/
LOCAL HANDLE NEAR setToScreen(HWND hWnd )
{HANDLE       hScreen;
 HWND         hWndScreen;
 hWndScreen   = GetWindow(hWnd,GW_CHILD);
 hScreen      = GetWindowWord(hWndScreen, GWW_SCREENHANDLE);
 sptr         = (NPHELPSCREEN)LocalLock( hScreen );
 return hScreen;
}
/*************************************************************************
```

```
 * FONCTION : differentScreen

 * PROPOS   : voit si hDlg se réfère a la même fenêtre que le dernier appel
*****************************************************************************/
LOCAL BOOL NEAR differentScreen(HWND hDlg)

{LOCAL  HWND   hWnd  = NULL;

 if (hWnd != hDlg)

    {hWnd = hDlg;

    return TRUE;
   }
 return FALSE;
}
/*****************************************************************************
 * FONCTION: initScreen
 * PROPOS  : prépare l'écran pour un message WM_CREATE.

*****************************************************************************/
LOCAL BOOL NEAR initScreen(HWND hWnd)
 {HANDLE hScreen;
  char   szMemErr[15];
  if (!(hScreen = LocalAlloc(LHND,sizeof(HELPSCREEN))))
    {LoadString(hInst,IDS_MEMERROR,szMemErr,15);
     MessageBox( GetParent(hWnd), szMemErr, NULL, MB_ICONEXCLAMATION);
     return FALSE;
    }
  sptr = (NPHELPSCREEN)LocalLock(hScreen);
  sptr->wScreen = lpsc->wScreen;
  sptr->wStart = lpsc->wStart;
  sptr->wEnd = lpsc->wEnd;
  LocalUnlock(hScreen);
  SetWindowWord( hWnd, GWW_SCREENHANDLE, hScreen );
  return TRUE;
 }
/*****************************************************************************
 * FONCTION : freeScreen
 * PROPOS   : On a terminé à l'écran, on libère alors la mémoire.

*****************************************************************************/
LOCAL VOID NEAR freeScreen(HANDLE hScreen)
{lpsc->wScreen = sptr->wScreen;
 lpsc->wEnd    = sptr->wEnd;
 lpsc->wStart  = sptr->wStart;
 LocalUnlock(hScreen);
 LocalFree(hScreen);
}
```

[LISTING CINQ]

```
/*
 * Initialisation en langage C des utilitaires de la bibliothèque
 */
#include <windows.h>
#include "helplib.h"
#include "prothelp.h"
/*****************************************************************************
 * prototypes de fonctions
 *****************************************************************************/
#define LOCAL  static
int          NEAR PASCAL LibInitC       ( HANDLE );
LOCAL BOOL   NEAR        registerWindow ( VOID );
/*****************************************************************************
 * variables globales
 *****************************************************************************/
HANDLE       hInst;
/*****************************************************************************
 * C est toute de suite appelé depuis le point d'entrée en assembleur.
 * Il faut que la bibliothèque ait un seul segment de données. Cf libinita.asm.
 * Le segment de données peut (devrait) être déplaçable.
 *****************************************************************************/
int NEAR PASCAL LibInitC(HANDLE hInstance)
{
    hInst = hInstance;


       return(registerWindow());

}
/*****************************************************************************/
LOCAL BOOL  NEAR registerWindow ( )
{WNDCLASS WndClass;          /* structure de la classe Window */
 WndClass.style           = CS_HREDRAW | CS_VREDRAW;
 WndClass.lpfnWndProc     = ScreenWndProc;
 WndClass.cbClsExtra      = 0;
 WndClass.cbWndExtra      = sizeof(HANDLE);
 WndClass.hInstance       = hInst;
 WndClass.hIcon           = NULL;
 WndClass.hCursor         = LoadCursor(NULL, IDC_ARROW);
 WndClass.hbrBackground   = COLOR_WINDOW+1;
 WndClass.lpszMenuName    = NULL;


 WndClass.lpszClassName    = "Screen";

 return RegisterClass(&WndClass);
}
```

[LISTING SIX]

```
; LIBINITA.ASM -  Définit les points d'entrée et accomplit l'initialisation
;                 pour les bibliothèques ayant leurs propores segments.
        TITLE   LIBINITA
?PLM = 1
?WIN = 1
memM = 1

.xlist
include cmacros.inc
.list

externFP    <LocalInit>
externFP    <UnlockSegment>
externNP    <LibInitC>
;
;   cx = taille du tas, comme défini dans les fichiers .def
;   di = "Instance handle". C'est la hInstance en C passée à WinMain.
;        NOTE: Pour une bibliothèque WINDOWS, hModule est interchangeable avec
;        hInstance.
;        C'est un magasin d'objets globaux contenant le segment de données
;        s'il y en a un. S'il n'y en a pas, c'est qu'il s'agit du magasin de
;        modules, lequel est aussi un pointeur visant le module dès qu'il
;        s'agit d'un objet global fixe.
;        NOTE: La raison et le contenu de hInstance ne sont pas documentés
;        et on ne doit pas trop s'y fier. En fait, on peut tenir compte de
;        ce que le contenu de di peut être passé à toute procédure attendant
;        quelque chose de hInstance.
;        Compter sur autre chose est TRES dangereux, le contenu de hInstance
;        pouvant être modifié dans de futures versions.
;        ds = segment de données pour votre tas.
;        es:si = pointeur visant la ligne de commande

_INIT SEGMENT BYTE PUBLIC   'CODE'
assume CS: _INIT            ; ???? assume vs assumes? ??????????????????
assumes DS,NOTHING          ; ???? assume vs assumes? ??????????????????
assumes ES,NOTHING          ; ???? assume vs assumes? ??????????????????

cProc   LibInitA, <FAR, PUBLIC, NODATA>, <si,di>
cBegin
    xor     ax,ax                   ; Renvoie la faute s'il n'y a pas de tas
    jcxz    ourexit
    cCall   LocalInit,<ds,ax,cx>    ; définit votre DS pour l'exécution de LocalAllocs.

    or      ax,ax
    jz      ourexit

    cCall   LibInitC,<di>           ; Procède à toute initialisation C
                                    ; di = hInstance.
    push    ax                      ; Sauve le résultat.
    mov     ax,-1


    cCall     UnlockSegment,<ax>      ; NOTE on déverrouille DS.

                                      ; Cela implique que l'on doit se servir de
                                      ; Lock/UnlockData chaque fois qu'on entre,
                                      ; de même pour toute procédure accédant à notre
                                      ; segment de données.


     pop     bx


    or      ax,ax                   ; Vérifie qu'aucun n'est défectueux.
    jz      ourexit
    or      bx,bx
    jnz     ourexit
    xor     ax,ax

ourexit:
cEnd
_INIT ENDS
end LibInitA
```

[LISTING SEPT]

```
LIBRARY      helplib
DESCRIPTION  'Copyright 1988, mkj'


STUB              'WINSTUB.EXE'

CODE         LOADONCALL   MOVEABLE  DISCARDABLE
DATA         MOVEABLE     SINGLE
HEAPSIZE     2048
CODE         MOVEABLE


SEGMENTS     _INIT      PRELOAD        MOVEABLE          DISCARDABLE

             HELPLIB    MOVEABLE       LOADONCALL    DISCARDABLE

EXPORTS
       Screen        @1


            Topics        @2

       ScreenDlgProc @3
       ScreenWndProc @4


            TopicsDlgProc @5
```

[LISTING HUIT]

```
#
# FICHIER: helplib
# PROPOS: make file for the helplib DLL
#
COMP  =  -c -Alnw  -Gsw -Zp -Os -FPa -W2 -D LINT_ARGS
OBJS  = helplib.obj helpdlg.obj libinita.obj libinitc.obj helplib.res
LOBJS = helplib+helpdlg+libinita+libinitc

.rc.res:
  rc -r $*.rc

.asm.obj:
  masm $*;

helplib.res:        helplib.rc helplib.h vsruntim.asc caveats.asc \
                    cookbook.asc ref.asc help.dlg topics.dlg

helplib.obj:        helplib.c helplib.h prothelp.h
    cl $(COMP) -NT $* $*.c

helpdlg.obj:        helpdlg.c helplib.h prothelp.h
    cl $(COMP) -NT HELPLIB $*.c

libinita.obj:       libinita.asm
    masm $*;

libinitc.obj:               libinitc.c
    cl $(COMP) -NT _INIT $*.c

helplib.exe: $(OBJS) helplib.def
     link4 $(LOBJS),helplib.exe/align:16,/map/li,mwinlibc mlibw mlibca   /NOE
,helplib.def
    rc      helplib.res helplib.exe
    implib helplib.lib helplib.def
```

[LISTING NEUF]

```
 /*********************************************************************
 * FICHIER: helpdemo.c
 * PROPOS : montrer comment on se sert de la DLL helplib
 *********************************************************************/
#include <windows.h>
#include "helpdemo.h"

HANDLE        hInstance;          /* Le gestionnaire d'Instances */
char          szClass[10];        /* nom de classe de la fenêtre (cf fichier .rc) */
char          szTitle[40];        /* titre de la fenêtre (cf ibidem) */
char          szAbout[40];        /* Chaîne de la boîte About (cf ibidem) */

static HWND   hWnd;

long FAR PASCAL WndProc (HWND, unsigned, WORD, LONG);
BOOL NEAR       Initialize( HANDLE, int );
 /*********************************************************************
 *Programme principale de l'application
 *********************************************************************/
int PASCAL WinMain( hInst, hPrevInst, lpszCmdLine, nCmdShow )
     HANDLE     hInst;          /* notre gestion d'instances */
     HANDLE     hPrevInst;      /* précédente instance de cette application */
     LPSTR      lpszCmdLine;    /* pointe sur tous paramètres de la ligne de commande */
     int        nCmdShow;       /* Paramètre à utiliser pour le premier ShowWindow */
     {
      MSG     msg;              /* structure des messages */
      HANDLE     hAccel;        /* Accelerateur */

      /* Sauve notre  magasin d'instances dans une variable globale */

      hInstance = hInst;
      /* Initialise l'application, quitte en cas d'erreur */
      if( ! Initialize( hPrevInst, nCmdShow ) )
                {return FALSE;
                }
      /* Boucle principale de traitement des messages. Prend chaque message, interprète les messages
       * de clavier et, enfin, redistribue chaque message dans la fenêtre idoine.
       */
       hAccel = LoadAccelerators(hInstance,szClass);
       while( GetMessage( &msg, NULL, 0, 0 ) )
            {if (!TranslateAccelerator(hWnd, hAccel, &msg))
                 {TranslateMessage( &msg );
                  DispatchMessage( &msg );
                 }

                }

      return msg.wParam;
     }
/*********************************************************************
*Initialiser l'application.
*********************************************************************/
BOOL NEAR Initialize( hPrevInst, nCmdShow )
     HANDLE     hPrevInst;          /* Précédente instance, 0 si c'est la première */
     int     nCmdShow;              /* Parametre provenant de WinMain pour ShowWindow */
     {
      WNDCLASS     WndClass;       /* structure de Classe pour RegisterClass */
      HMENU        hMenu;          /* Contient le menu (système)  */
      if( ! hPrevInst ) {
           /* Initialisation de la première instance seulement */
           /* Charge les chaînes depuis le fichier-ressource */
           LoadString( hInstance, IDS_CLASS,     szClass,     sizeof(szClass) );
           LoadString( hInstance, IDS_TITLE,     szTitle,     sizeof(szTitle) );
           /* Prend note de notre classe fenêtre */
           WndClass.style             = CS_HREDRAW | CS_VREDRAW;
           WndClass.lpfnWndProc       = WndProc;
           WndClass.cbClsExtra        = 0;

                   WndClass.cbWndExtra        = 0;

           WndClass.hInstance         = hInstance;
           WndClass.hIcon             = LoadIcon( NULL, IDI_APPLICATION );
           WndClass.hCursor           = LoadCursor( NULL, IDC_ARROW );
           WndClass.hbrBackground     = COLOR_WINDOW + 1;
           WndClass.lpszMenuName      = szClass;
           WndClass.lpszClassName     = szClass;
           if( ! RegisterClass( &WndClass ) )
               return FALSE;
          }
     else
        {/* Initialisation pour les prochaines instances seulement */
         /* Copie les données de l'instance précédente */
         GetInstanceData( hPrevInst, szClass,     sizeof(szClass) );
         GetInstanceData( hPrevInst, szTitle,     sizeof(szTitle) );
        }
     /* Initialisation pour toute instance */

     /* Crée la fenêtre  */

         hWnd = CreateWindow(

          szClass,                                  /* nom de classe */
          szTitle,                                  /* titre de fenêtre */
          WS_OVERLAPPEDWINDOW                       /* style de fenêtre */
          CW_USEDEFAULT,                            /* x */
          0,                                        /* y */
          CW_USEDEFAULT,                            /* largeur x */

                  0,                /* largeur y */

          NULL,               /* Parent hWnd (aucun si au sommet) */
          NULL,               /* Gestion du Menu */
          hInstance,          /* Gestion d'instance */
          NULL                /* Paramètre à passer dans WM_CREATE  (none) */
      );


/* Insère "About..." dans  le menu système */

      LoadString( hInstance, IDS_ABOUT,     szAbout,     sizeof(szAbout) );
      hMenu = GetSystemMenu(hWnd, FALSE);

      ChangeMenu(hMenu, 0, NULL, 999, MF_APPEND |  MF_SEPARATOR);

      ChangeMenu(hMenu, 0, (LPSTR)szAbout, IDS_ABOUT, MF_APPEND | MF_STRING);

      /* rend la fenêtre visible  */

      ShowWindow( hWnd,  nCmdShow );

      /* On a toutes les informations, mettons à jour notre affichage */
      UpdateWindow( hWnd );

      return TRUE;

       }
```

[LISTING DIX]

```
 /*********************************************************************
 * FICHIER: wndproc.c
 * PROPOS : Les fonctions WndProc et About pour la démo de la DLL helplib.
 *********************************************************************/
#include "windows.h"
#include "helpdemo.h"
extern HANDLE hInstance;
 /****************** Prototype de fonction Message ****************/
extern   VOID   HelpDemoMsg  ( HWND, WORD, LONG );
 /****************** prototype de fonction locale *****************/
BOOL NEAR           paint  ( HWND );

long FAR PASCAL     WndProc( HWND,  unsigned, WORD, LONG );


BOOL FAR PASCAL        AboutDlgProc( HWND, unsigned, WORD, LONG );

long FAR PASCAL WndProc( hWnd, message, wParam, lParam )
     HWND hWnd;
     unsigned message;
     WORD wParam;
     LONG lParam;
     {
      FARPROC lpprocAbout;
      switch (message)
           {
            case WM_SYSCOMMAND:
```

```
                switch (wParam)
                {
                 case IDS_ABOUT:
                      /* Bind callback function with module instance */
                      lpprocAbout = MakeProcInstance( (FARPROC)AboutDlgProc,
                                                      hInstance);
                      DialogBox( hInstance, MAKEINTRESOURCE(ABOUTBOX), hWnd,
                                                      lpprocAbout );
                      FreeProcInstance( (FARPROC)AboutDlgProc );
                      break;
                 default:
                      return DefWindowProc( hWnd, message, wParam, lParam );
                      }
                break;
           case WM_COMMAND:
                switch(wParam)
                {


                              case IDM_HELP:


                     HelpDemoMsg (hWnd, wParam, lParam);
                     break;
                }
           case WM_PAINT:
                paint( hWnd );
                break;
           case WM_DESTROY:
           case WM_CLOSE:
                PostQuitMessage( 0 );
                break;
           default:
                return DefWindowProc( hWnd, message, wParam, lParam );
                break;
         }
     return(0L);
    }
 /************************************************************************
  * procedure de coloriage (pour traiter les messages WM_PAINT )
  ************************************************************************/
BOOL   NEAR  paint ( hWnd )
     HWND hWnd;
     {
      PAINTSTRUCT ps;
      BeginPaint(hWnd,&ps);


      EndPaint(hWnd,&ps);



         return TRUE;


     }
 /************************************************************************
  * boîte About
  ************************************************************************/
BOOL FAR PASCAL AboutDlgProc( hDlg, message, wParam, lParam )
     HWND hDlg;
     unsigned message;
     WORD wParam;
     LONG lParam;
     {
      if (message == WM_COMMAND)
           {EndDialog( hDlg, TRUE );
            return TRUE;
           }
      else if (message == WM_INITDIALOG)
           return TRUE;
      else return FALSE;
     }


[LISTING ONZE]
/************************************************************************
 * Les fonctions de messagerie
 ************************************************************************/
#include "windows.h"
#include "helpdemo.h"
#include "helplib.h"          /* fichier d'assistance de la bibliothèque */
#include "prothelp.h"          /* prototypes des fonctions de la bibliothèque
*/

extern HANDLE hInstance;

 /****************** prototypes des fonctions locales ************/
VOID   HelpDemoMsg  ( HWND, WORD, LONG );



 /************************************************************************/

VOID    HelpDemoMsg( hWnd, wParam, lParam )
     HWND hWnd;
     WORD wParam;
     LONG lParam;
    {SCREEN sc;
     sc.wStart = VSRUNTIME;
     sc.wEnd = REF;
     Topics(&sc);
     }

[LISTING DOUZE]

/************************************************************************
```

```
 * FICHIER: prothelp.h
 * PROPOS : prototypes de fonctions pour la DLL helplib
 ************************************************************************/
#ifndef   HELPDLG
  #define HELPDLG extern
#endif
#ifndef   HELPLIB
  #define HELPLIB extern
#endif

HELPLIB BOOL  FAR PASCAL  Screen          ( LPSCREEN );
HELPLIB VOID  FAR PASCAL  Topics          ( LPSCREEN );


HELPDLG BOOL  FAR PASCAL  ScreenDlgProc     ( HWND, WORD, WORD, LONG );

HELPDLG BOOL  FAR PASCAL  TopicsDlgProc   ( HWND, WORD, WORD, LONG );
HELPDLG LONG  FAR PASCAL  ScreenWndProc   ( HWND, WORD, WORD, LONG );
extern LPSTR  FAR PASCAL  lstrcpy         ( LPSTR, LPSTR );
extern int    FAR PASCAL  lstrlen         ( LPSTR );


[LISTING TREIZE]
/************************************************************************
 * FICHIER: helpdemo.h
 * PROPOS :fichier d'inclusion pour la fenêtre de démonstration de la DLL d'assistance
 ************************************************************************/
/************** strings ************************************************/
#define    IDS_CLASS              0  /* ID de la table de noms de classes */
#define    IDS_TITLE              1  /* ID de la Table de Titres */


#define    IDS_ABOUT              2  /* ID de la table de libellés de la boîte About  */


/************** menus **************************************************/
#define     ABOUTBOX          3    /* ID de la ressource "dialogue" de la boîte About */


#define    IDM_HELP           1000   /* ID de la  ressource "Menu" */



[LISTING QUATORZE]

NAME         HELPDEMO
DESCRIPTION  'Help DLL Demonstration, Copyright 1988, mkj'
STUB         'WINSTUB.EXE'
CODE         LOADONCALL  MOVEABLE  DISCARDABLE


DATA         MOVEABLE     MULTIPLE

HEAPSIZE     2048
STACKSIZE    4096
EXPORTS
       WndProc         @1
       AboutDlgProc    @2

[LISTING QUINZE]
/************************************************************************
 * FICHIER: helpdemo.rc
 * PROPOS : utilisé avec la DLL helplib pour démonstration
 ************************************************************************/
#include <style.h>
#include "helpdemo.h"

STRINGTABLE
BEGIN
     IDS_CLASS           "HelpDemo"
     IDS_TITLE           "Help Demonstration"
     IDS_ABOUT           "About..."
END

ABOUTBOX DIALOG 22, 17, 154, 75
STYLE WS_POPUP | WS_DLGFRAME
BEGIN
     CTEXT "DLL Example"                      -1, 0,  5, 154, 8
     CTEXT "Help Library Demonstration"       -1, 0, 14, 154, 8
     CTEXT "Version 1.00"                     -1,30, 34,  94, 8
     CTEXT "Copyright ) 1988, mkj"            -1, 0, 47, 154, 9
     DEFPUSHBUTTON "Ok"                  IDOK, 61,59, 32,  14, WS_GROUP
END

HelpDemo   MENU
BEGIN
  MENUITEM "\aF1=Help",                  IDM_HELP,HELP
END

HelpDemo  ACCELERATORS
BEGIN
   VK_F1,    IDM_HELP, VIRTKEY
END

[LISTING SEIZE]

TOPICS_BOX DIALOG LOADONCALL MOVEABLE DISCARDABLE 28, 19, 206, 79
STYLE  WS_DLGFRAME | WS_POPUP
BEGIN
     CONTROL "", ID_LB_TOPICS, "listbox", LBS_NOTIFY | WS_BORDER | WS_VSCROLL
| WS_CHILD | TABGRP, 12, 18, 144, 49
     CONTROL "&Help", IDOK, "button", BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD
| TABGRP, 165, 17, 32, 14
      CONTROL "Cancel", IDCANCEL, "button", BS_PUSHBUTTON  | WS_TABSTOP  |
WS_CHILD , 165, 43, 32, 14
```

```
    CONTROL "Help Topics on DLL's", 103, "static", SS_LEFT | WS_CHILD, 12, 5,
100, 11
END
```

[LISTING DIX-SEPT]

```
HELP_BOX DIALOG LOADONCALL MOVEABLE DISCARDABLE 10, 9, 262, 131
STYLE WS_BORDER | WS_CAPTION | WS_POPUP | WS_SYSMENU | WS_VISIBLE
CAPTION  "Help"
BEGIN
CONTROL "", ID_SCREEN_HELP, "Screen",  WS_BORDER | WS_CHILD | TABGRP, 9, 9, 234, 90
CONTROL "", ID_SCROLL_HELP, "scrollbar", SBS_VERT | WS_CHILD | WS_CLIPSIBLINGS, 242, 9, 8, 90
CONTROL "&Next", ID_NEXT_HELP, "button", BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD, 76, 105, 44, 16
CONTROL "&Previous", ID_PREVIOUS_HELP, "button", BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD, 139, 105, 44, 16
CONTROL "Cancel", IDCANCEL, "button", BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD, 201, 105, 44, 16
CONTROL "&Topics", ID_TOPICS_HELP, "button", BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD, 14, 105, 44, 16
END
```

[LISTING DIX-HUIT]

```
#
# FICHIER: helpdemo
# PROPOS : crée le fichier pour la démonstration de la DLL helplib
#
COMP  = -c -AS -D LINT_ARGS -Os -Zp  -Gsw -FPa
ASM   =

.rc.res:
  rc -r $*.rc

.c.obj:
  cl $(COMP) $*.c

.asm.obj:
  masm $(ASM) $*;

helpdemo.res:    helpdemo.rc  helpdemo.h

helpdemo.obj:    helpdemo.c   helpdemo.h

wndproc.obj:     wndproc.c    helpdemo.h

msgs.obj:        msgs.c       helpdemo.h helplib.h prothelp.h

helpdemo.exe:    helpdemo.obj wndproc.obj msgs.obj helpdemo.res
 link4  helpdemo+wndproc+msgs,/align:16,/map/li,slibw slibca helplib /NOE /co
,helpdemo.def
 rc helpdemo.res
```

[LISTING DIX-NEUF]

```
/*   ************************************
     FICHIER: helplib.act
     PROPOS : exemple de désignation et de chargement de
         la bibliothèque à l'exécution au lieu que ce soit lors de
         l'édition des liens (link4). Le code est écrit au moyen d'Actor,
         un langage de programmation orientée objets dédié au développement
         d'applications pour MS-WINDOWS.
     ************************************ */

/*   ************************************
     INITIALISATION
     ************************************ */
/* define constants */
#define VSRUNTIME       300
#define CAVEATS         301
#define COOKBOOK        302
#define REF             303

#define A               500
#define B               501
#define C               502
!!

/* crée une nouvelle struture C */
SCREEN := new(CStruct);
!!

init(SCREEN, #(
  int wScreen 1
  int wStart  1
  int wEnd    1
));
!!
/* initialise la bibliothèque d'assistance */
HelpLib := new(Library);            /* crée une nouvelle bibliothèque   */
HelpLib.name := "HELPLIB.EXE";      /* fixe le nom de fichier de la bibliothèque */
add(HelpLib, #SCREEN, 0, #(1));     /* donne les noms des fonctions exportées
de la bibliothèque, auxquelles vous ferez appel.  '0' est le mode de retour
(0=int,1=long), et '#(1)' inventorieles arguments - une variable "long"*/
!!
/* charge la bibliothèque */
load(HelpLib);
!!
/*   ************************************
     BODY
     ************************************ */
/* remplit la structure C */
SCREEN[#wStart] := VSRUNTIME;
SCREEN[#wEnd] := REF;
SCREEN[#wScreen] := CAVEATS;
!!
/* appelle la DLL pour faire monter le dialogue d'assistance */
pcall(HelpLib.procs[#SCREEN], lP(SCREEN));
freeHandle(SCREEN);
!!
/*   ************************************
     CLEAN-UP
     ************************************ */
/* libère la bibliothèque */
free(HelpLib);
!!
```


N°1

1 MARS 1989

60.000 EXEMPLAIRES

DEBUG

3, rue l'Olive

75018 PARIS

Plaquette distribuée lors du PC Forum 1989

DEBUG MAGAZINE

CHERCHE LECTEURS FIDELES

DEBUG MAGAZINE recherche ses lecteurs le plus fidèles, ceux qui nous connaissent depuis « toujours ».
Qui se rappelle du lancement du magazine lors du FORUM PC 1989 ?
Parmi ceux qui vont nous rapporter la preuve (par exemple le dépliant que nous avons distribué dans les allées), nous allons tirer au sort trois logiciels exceptionnels :

1. FoxBase Plus 2.10
2. Rbase Microsoft
3. Sprint de Borland

10. Guides Norton pour C, Pascal, Basic, ASM et OS/2 et 10 Utilitaires Norton Avancés 4.5

LA RECURSIVITE DEMYSTIFIEE...

LA RECURSIVITE DEMYSTIFIEE...

par **Jean-Christophe RIAT**

Qu'est-ce que la récursivité? Le dictionnaire donne la définition suivante: « la récursivité est la propriété de ce qui est récursif »! A côté du terme récursif est écrit: « se dit de quelque chose qui peut être répété de façon infinie ». Cette définition est certainement très correcte pour les usages courants, mais elle tenterait à laisser penser que les académiciens ne pratiquent pas l'informatique...

Pour cette discipline, une fonction ou une procédure est dite récursive, si elle fait appel à elle-même dans sa propre définition. Cette notion est magnifiquement illustrée par la lithographie intitulée « Mains dessinant » du graveur contemporain hollandais M.C. Escher. Elle représente une feuille de papier sur laquelle sont tracées deux mains munies chacune d'un crayon; celles-ci se dessinent mutuellement. Le dessin est donc bien récursif puisqu'il se définit lui-même: les mains se créent elles-mêmes!

Un cas d'école: la factorielle...

Pour illustrer cette définition, commençons par la star de la récursivité, la plus célèbre des fonctions récursives: la factorielle. Appliquée à un nombre entier n, cette fonction retourne la valeur du produit de n par n-1, par n-2 . . . jusqu'à 2 puis 1. Avec les notations mathématiques:
n! = n x (n-1) x (n-2) x . . . x 2 x 1.

Cette expression correspond au nombre de groupes différents que l'on peut former avec n éléments distincts en tenant compte de l'ordre de ces éléments. Ainsi il y a 5!, soit 120, manières de distribuer cinq prix littéraires entre cinq éditeurs, si on suppose que chaque éditeur doit recevoir un prix...

Pour programmer la fonction factorielle, la première idée qui vient à l'esprit est d'utiliser la structure de contrôle FOR . . . DO . . .
Un exemple de solution est alors:

```
FUNCTION Factorielle (n : INTEGER) : INTEGER;
   VAR i : INTEGER;
   BEGIN
      Factorielle:=1;
    FOR i:=1 TO n DO Factorielle:=i x Factorielle;
   END;
```

Cette façon de résoudre le problème est tout à fait correcte. Mais, en remarquant que n! = n x (n-1)!, une autre solution, récursive cette fois-ci, peut être envisagée. Cette relation peut être considérée comme une définition de la fonction factorielle, définition o• apparaît la dite factorielle... Mais cette définition reste incomplète: il manque une valeur d'initialisation. Celle-ci est toute trouvée si on sait que 0! est égal à 1. Avec cette valeur, et la relation donnée ci-dessus, il devient possible de calculer, de proche en proche, la factorielle de tout entier n. Ainsi pour trouver 4!, on a les relations:

RECURSIVITE...
Art ultime de la programmation pour certains, sujet tabou pour d'autres! Beaucoup préfèrent développer leur application avec des « solutions éprouvées », plutôt que de se laisser séduire par l'univers merveilleux des fonctions récursives... Merveilleux car, comme nous allons l'illustrer avec le langage Pascal, Fée Récursivité résout certains problèmes d'une manière très élégante, presque par magie...

```
4! = 4 x 3!
3! = 3 x 2!
2! = 2 x 1!
1! = 1 x 0! = 1 x 1 = 1

En remontant, on obtient alors:
2! = 2 x 1 = 2, puis 3! = 3 x 2 = 6 et finalement 4! = 4 x 6 = 24.
```

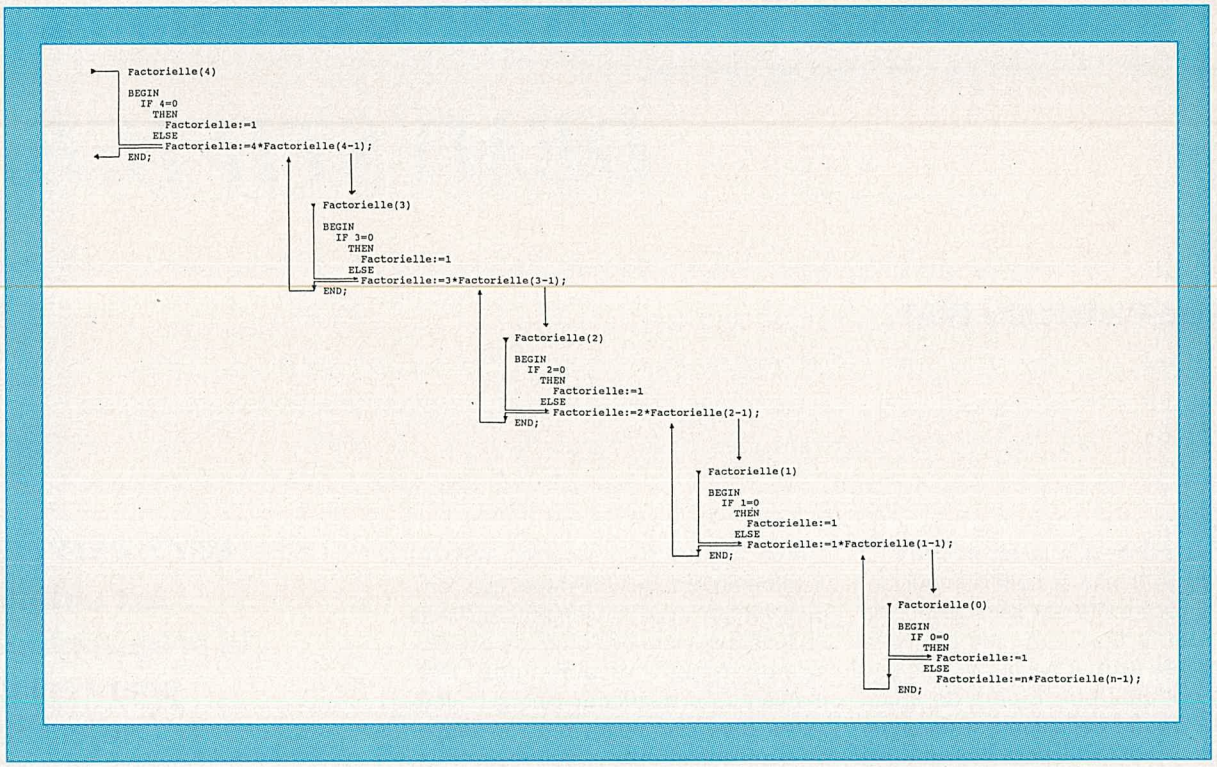


Figure 1 : ***Cheminement pour le calcul de Factorielle (4)***

Mais comment utiliser, de façon simple, ces remarques pour programmer la fonction factorielle? C'est ici que la récursivité apporte son caractère magique. Il suffit en effet de traduire en langage informatique la relation et l'initialisation exposées précédemment. Et c'est tout! Une solution récursive est alors: (encadré ci-après)

est appelée, avec cette fois-ci n-1 comme valeur de paramètre. Lors du retour de celle-ci, avec comme résultat (n-1)!, le contexte est restitué et l'expression « n x Factorielle (n-1) » évaluée. Le résultat final de Factorielle (n) peut alors être retourné. Ce mode de fonctionnement génère des appels successifs de la même fonction Factorielle, avec des valeurs de paramètre qui décroissent jusqu'à zéro. A titre d'illustration, la figure 1 décrit le cheminement du pointeur de programme pour calculer Factorielle (4).

Des fonctions statistiques récursives...

La factorielle est la plus simple des fonctions récursives. Elle utilise un unique paramètre d'appel (la variable n). La récursivité est à un seul niveau, puisque la fonction apparaît une seule fois dans le membre de droite de la relation de récurrence. La condition d'arrêt est unique. C'est un excellent« cas d'école » pour comprendre les principes de la récursivité. Il ne montre cependant que partiellement la puissance de cette façon de programmer. La fonction mathématique de cal-

```
FUNCTION Factorielle (n : INTEGER) : INTEGER;
BEGIN
   IF n = 0
        THEN Factorielle := 1
        ELSE Factorielle := n x Factorielle (n-1);
END;
```

Lors de l'exécution de cette fonction, le calculateur évalue l'expression booléenne « n = 0 ». Si le test est vrai, la valeur 1 est retournée. Dans le cas contraire, le calcul du résultat nécessite de connaître « Factorielle (n-1) ». Le contexte est alors sauvegardé dans la pile et une nouvelle fonction Factorielle

■

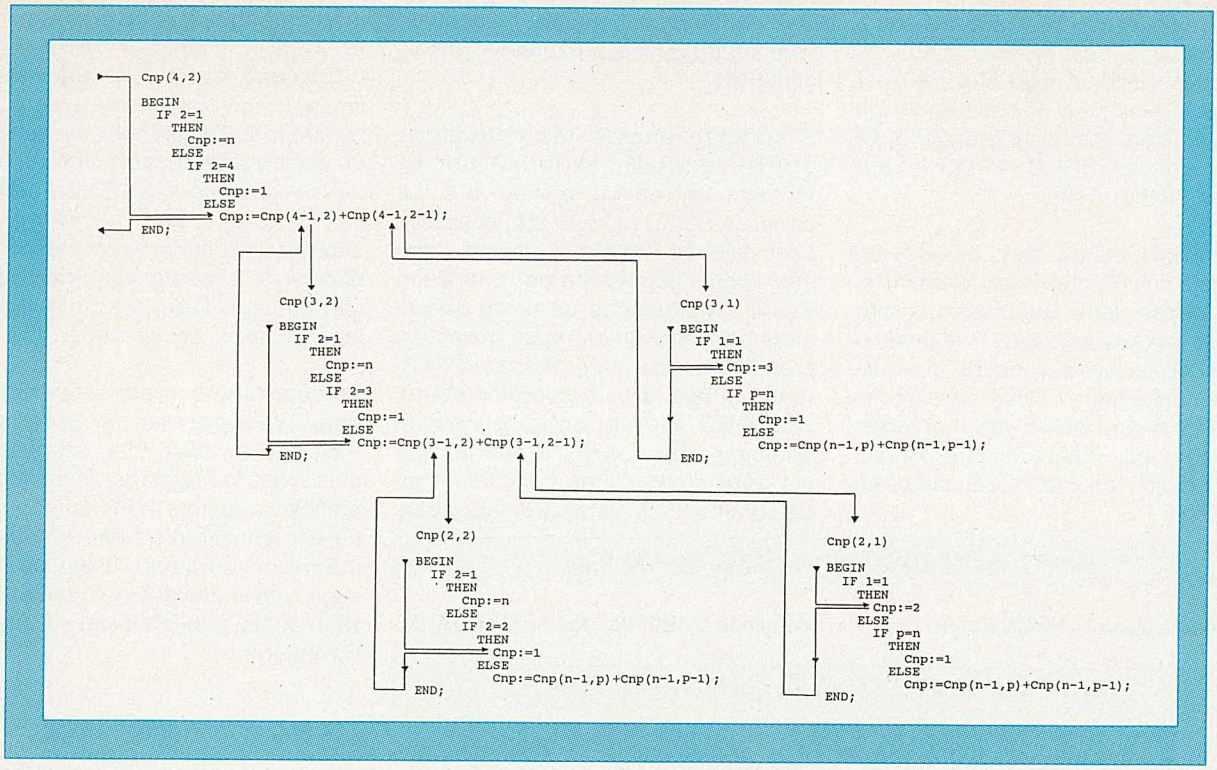


Figure 2 : ***Cheminement pour le calcul de*** C_4^2

cul de combinaisons Cnp est beaucoup plus intéressante. Avec les notations mathématiques, Cnp = n! / (p! x (n-p)!) Cette valeur correspond au nombre de groupes que l'on peut former avec p éléments distincts pris parmi n, sans tenir compte de l'ordre des éléments. Ainsi, il existe C105, soit 252, façons de choisir cinq éditeurs parmi dix pour leur attribuer les cinq prix littéraires précédents...

Pour mettre en oeuvre cette fonction, une première solution est de la calculer avec sa définition, en utilisant la factorielle définie précédemment. Cette solution est mauvaise car les calculs dépassent rapidement la précision numérique de l'ordinateur: les résultats intermédiaires de n! et p! donnent des nombres trop grands, qui sont souvent arrondis avec une représentation de type réel. La relation du triangle de Pascal va être très utile: Cnp = Cn-1p + Cn-1p-1. Celle-ci fait immédiatement penser à une solution récursive. Mais au lieu de n'avoir qu'un seul niveau de récursivité, comme pour la factorielle, il y en a ici deux: la fonction Cnp apparaît deux fois dans sa propre définition!

Cela n'est pas gênant: comme pour la factorielle, la programmation récursive de la fonction Cnp consiste simplement à traduire la relation récurrente en langage informatique. Toute la récursivité est gérée automatiquement par le programme. Il faut cependant faire attention aux conditions d'arrêt. Les appels successifs de la fonction Cnp font décroître dans un cas le paramètre n seul (pour le terme Cn-1p), et dans l'autre cas les deux paramètres n et p (pour le terme Cn-1p-1). Il faut donc deux conditions d'arrêt. Puisque n est supérieur à p, diminuer sa valeur de un à chaque appel implique que n-1 atteigne la valeur de p. Une première condition d'arrêt est donc fournie par la valeur de Cpp qui est 1. De même, diminuer p de un à chaque récurrence entraîne que p-1 atteigne la valeur 1. La seconde condition d'arrêt est alors donnée par Cn1 qui vaut n.

Voilà, ce n'est pas plus compliqué! La programmation de Cnp se fait alors tout naturellement, comme pour la factorielle. Un exemple de mise en oeuvre en Pascal est fourni sur le listing ci-joint. La figure 2 décrit le cheminement du pointeur de programme pour calculer C42. On y voit clairement la double récursivité.

Une autre fonction intéressante à programmer de manière récursive est le calcul d'arrangements Anp. Avec les notations mathématiques, Anp = n! / (n-p)! Cette valeur correspond au nombre de groupes que l'on peut former avec p éléments distincts pris parmi n, en tenant compte de l'ordre des éléments. Ainsi, il existe A105, soit 30240, manières d'attribuer cinq prix littéraires entre dix éditeurs, si on suppose que chaque éditeur peut recevoir au plus un prix...

En remarquant que Anp = p! x Cnp, la fonction Anp pourrait se programmer à l'aide des deux fonctions précédemment détaillées. Mais une solution récursive est également possible grâce à la relation: Anp = An-1p + p x An-1p-1. Celle-ci est du même type que celle utilisée pour la fonction Cnp. La recherche des conditions d'arrêt conduit donc à la recherche des valeurs de App et de An1. Pour An1, aucune difficulté puisque An1 est égal à n. Pour App par contre, cela semble plus délicat puisque App = Factorielle (p). Mais Fée Récursivité va une fois encore venir à notre secours. Il suffit de transformer cette condition d'arrêt en un nouvel appel récursif par la relation: App = p x An-1p-1. Tous les éléments sont donc réunis pour programmer la fonction Anp de façon récursive. Un exemple de code Pascal est fourni sur le listing ci-joint.

Des fonctions arithmétiques récursives...

Pour terminer cette visite dans l'univers de la récursivité, examinons deux fonctions mathématiques bien connues par les collégiens: le PGCD et le PPCM. Le PGCD de deux entiers a et b est leur Plus Grand Commun Diviseur, c'est-à-dire le plus grand

nombre entier qui divise à la fois a et b. Le PPCM de deux entiers a et b est leur Plus Petit Commun Multiple c'est-à-dire le plus petit nombre entier qui est divisible à la fois par a et par b. On a par exemple PGCD (45,60) = 15 et PPCM (45,60) = 180.

Nos chères petites têtes blondes ont appris à calculer ces deux fonctions en développant leurs arguments en produit de facteurs premiers. Cette méthode est assez lourde à mettre en oeuvre par informatique... Mais la science mathématique recèle des merveilles! Les ouvrages d'arithmétique contiennent en effet, pour le PGCD, la formule suivante:

si a > b, alors PGCD (a,b) = PGCD (a MOD b, b).

De même, pour le PPCM on a:

si a > b, alors PPCM (a,b) = a / (a-b) PPCM(a-b,b).

Même si ces relations n'ont pas la même forme que celles rencontrées jusqu'à maintenant, elles vont permettre une programmation récursive. Comme elles ne sont valables que dans les cas où a est supérieur à b, il est nécessaire de savoir quoi faire dans les autres cas. Il est facile de voir que: PGCD (a,b) = PGCD (b,a) et de même que PPCM(a,b) = PPCM (b,a).

Ces relations, à nouveau récursives, complètent celles présentées précédemment.

Il manque encore les conditions d'arrêt. Le résultat de (a MOD b) est toujours strictement inférieur à b. Le calcul de PGCD (a MOD b,b) nécessite donc celui de PGCD(b,a MOD b). Lors des appels successifs de la fonction PGCD, le paramètre de droite décroît alors progressivement pour finalement prendre la valeur zéro. Puisque PGCD (a,0) est égal à a, la condition d'arrêt est PGCD (a,0)=a. Un raisonnement analogue pour le PPCM montre que les appels récurrents conduisent au calcul de PPCM (a,a) qui vaut a. La condition d'arrêt est donc PPCM (a,a) =a.

Avec ces éléments, les deux fonctions PGCD et PPCM se programment aisément de manière récursive. Un exemple de programmation en Pascal est donné sur le listing ci-joint. La figure 3 décrit le cheminement du pointeur programme pour le calcul de PGCD (60,45). Cette mise en oeuvre récursive est particulièrement élégante car elle ne fait qu'utiliser les propriétés mathématiques attachées aux fonctions. Le programmeur n'a rien d'autre à faire qu'à traduire ces propriétés en langage informatique sans se soucier comment le programme va les mettre en oeuvre à l'exécution. Voilà la magie de la récursivité évoquée au début de cet article... Sur le listing sont détaillées les cinq fonctions récursives décrites. Un programme principal permet de les utiliser, afin de vérifier que la récursivité... ça marche!

La récursivité n'est que très peu employée par les programmeurs Pascal, C ou ADA. Elle l'est nettement plus avec les langages d'intelligence artificielle comme LISP ou PROLOG, où elle constitue un des fondements même du langage. C'est pourquoi nous conclurons par la formule: « si la récursivité n'existait pas, il faudrait l'inventer » !.

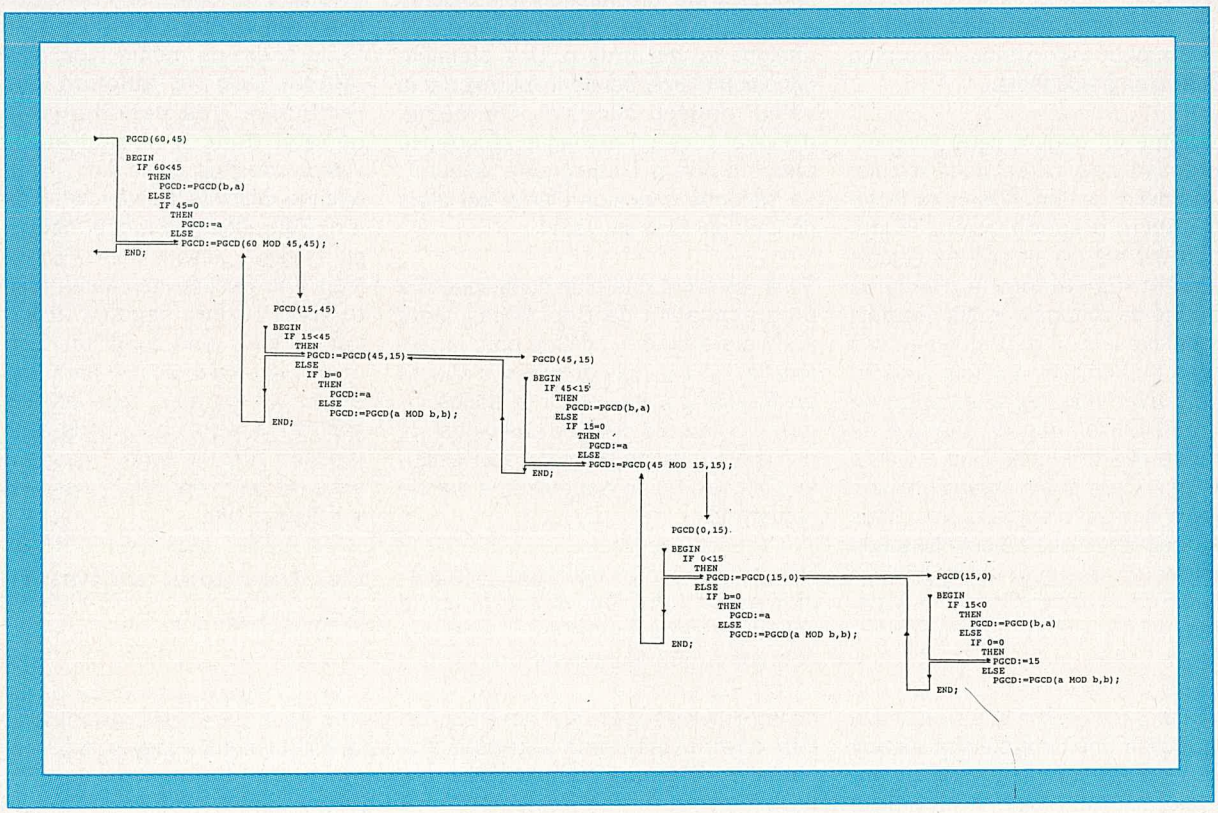


Figure 3 : ***Cheminement pour le calcul de PGC D (60,45)***

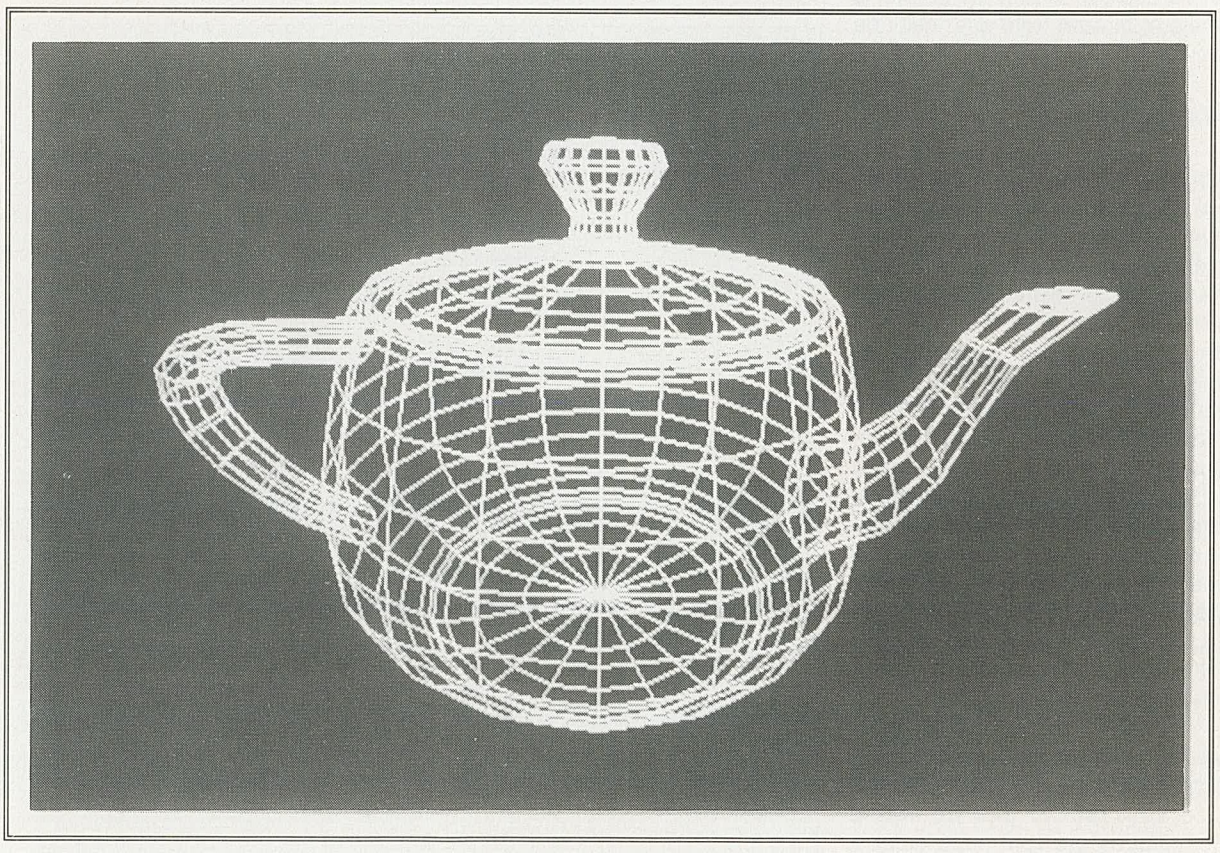
A CUP OF TEA ?

(Vous prendrez bien un peu de thé ?)

par **Jean-Pascal COTE**

Les origines de la théière

Dans les années 70 à l'université de l'UTAH, il y eut des recherches importantes sur la mise au point d'algorithmes de lissage et de mapping. Certe, des tests furent effectués sur des objets simples tels que des sphères, des cubes, et des cylindres, mais en fait les équipes de recherche désiraient travailler sur d'autres formes plus complexes. En 1971, Ivan Sutherland fit modéliser sa VW Coccinelle par ses élèves. On la vit donc circuler dans les rues de la ville pendant plusieurs semaines, bariolée de lignes et de repères utilisés pour l'établissement des points de contrôle.



Théière vue de côté

Un beau jour de 1975, Martin Newell remarqua les formes agréables de sa théière posée sur son bureau et décida à son tour de la modéliser par des surfaces Bezier. Il la décomposa en 5 parties : le bord, le couvercle, le bec, la poignée et le corps. Le fond ne fut rajouter que plus tard. Le bec et la poignée furent traités comme des cylindres extrudés tandis que le corps, le bord et le couvercle furent générés séparement, en considérant leurs profiles en rotation autour d'un axe. La taille des données fut quelque peu diminuée et modifiée pour la rendre plus esthétique en certains points. La première apparition de la théière eut lieu dans un article du SIGGRAPH de 1976. Mais sa célébrité ne vint que plus tard quand Jim Blinn testa ses algorithmes de mapping de textures donnant un regard nouveau sur cet objet quotidien. Par la suite elle devint un passage obligatoire pour les fournisseurs de librairies graphiques et connue dans le monde des stations de travail, le succès que l'on sait.

Aujourd'hui la théière de Martin Newell repose en paix au Computer Museum de Boston, après avoir engendré des milliers de petites soeurs. Elle se retrouve entourée d'objets beaucoup plus complexes qu'elle dans ce cimetière de l'informatique, parce qu' un homme a décidé, un jour, de donner à une théière un autre rôle que celui de verser le thé.

La modélisation

A l'origine la théière n'était composée que de 28 patchs (un patch représentant une surface Bezier élémentaire sous-tendue par 16 points de contrôle). Par la suite on lui rajouta 4 pachs supplémentaires modélisant le fond. Le fichier « patch.dat » rassemble les données constituant la théière. 4 patchs pour le bord, 8 pour le corps, 4 pour la poignée, 4 pour le bec, 8 pour le couvercle et 4 pour le fond, soient 32 en tout. Chacune des lignes, représentant un patch, renvoie aux numéro des 16 points de contrôle explicités dans le fichier « surface.dat ». Comme les patchs sont collés les uns aux autres, on retrouve donc des rangées de points identiques de façon à donner cette impression de continuté.

Il nous faut maintenant aborder la partie la plus délicate de cette modélisation, je veux parler des quelques mathématiques qui vont nous aider à comprendre le mécanisme de représentation.

Sans rentrer dans les détails résumons le fait qu'une surface Bezier peut s'écrire de façon matricielle :

$$M(u,v) = [u3,u2,u,1] * [Mb] * [Pij] * [Mb]t * [v3,v2,v,1]$$

où [Pij] est la matrice des 16 points de contrôle de coordonnées spaciales (X,Y,Z) , [Mb] la matrice des coefficients Bezier de la forme :

$$[Mb] = \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix}$$

et [Mb]t la matrice Bezier transposée. Les paramètres u et v varient chacun dans l'intervalle [0,1].

Pour tracer une surface maillée, il suffira donc de fixer u puis faire varier v, et de recommencer l'opération dans l'autre sens. La théière étant calculer en 3D, il nous a fallu lui ajouter un algorithme de projection ramenant à la fenêtre graphique de notre écran l'objet projeté. Ce genre d'algorithme étant classique, on ne s'y attardera pas, mais pour plus de détails on consultera [1].

Le pseudo programme principal aura donc la forme suivante :

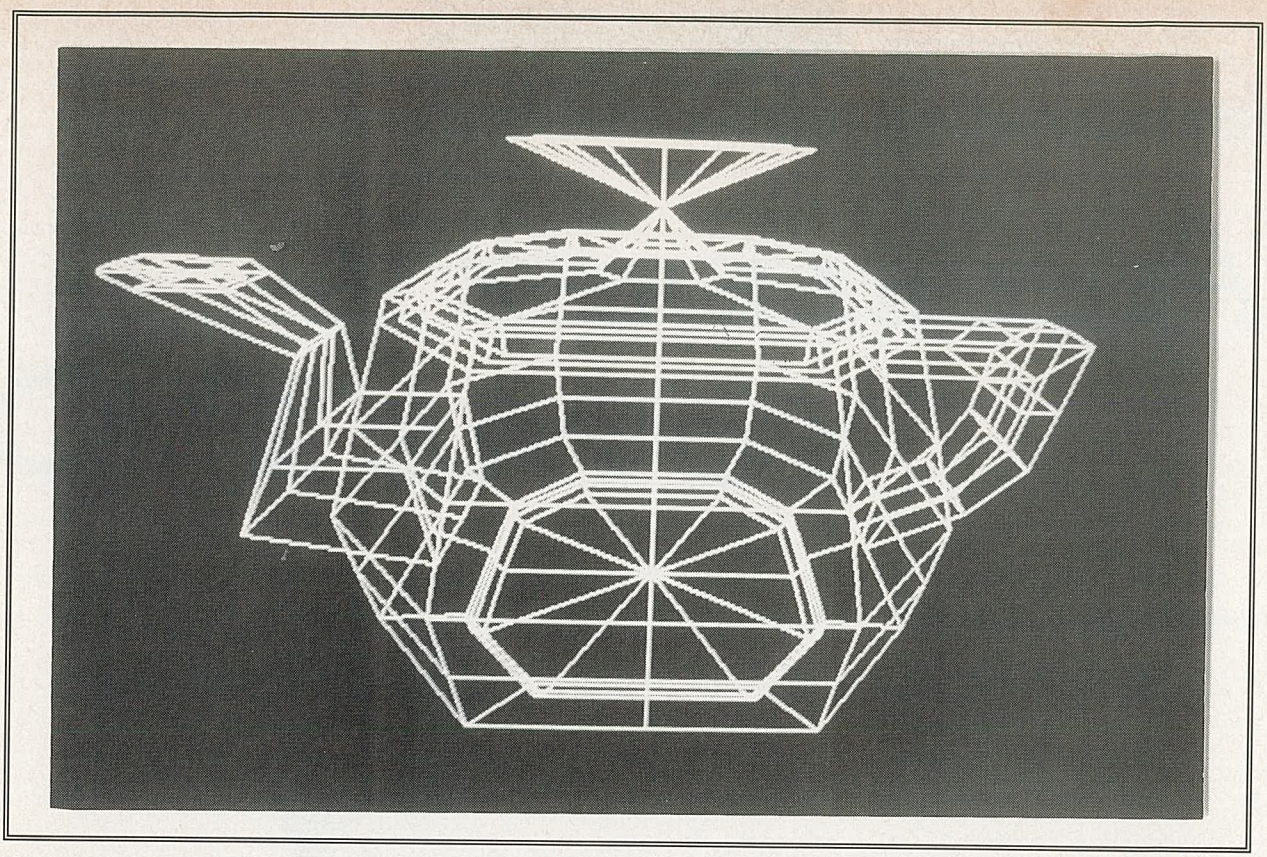
```
Calculer la matrice transposée [Mb]t
pour patch = 0 jusqu'a 31          (32 patchs)
début
    Lire des points de contrôle de la surface
    Pour chaque composant X, Y, Z effectuer le produit matriciel

        [A] = [Mb] * [Pij] * [Mb]t
        /* Tracé des ISO_U */
        Pour maillage en u = 0 jusqu'a 1, pas de  (u/pas_u)
        début
            faire le produit   [K] = [u3,u2,u,1] * [A]
            pour v = 0 jusqu'a 1, pas de  (v/pas_v)
            début
                faire le produit   [K] * [v3,v2,v,1]t
                Projeter le résultat 3D en 2D
                Tracer
            fin
        fin
        /* Tracé des ISO_V */
        pour maillage en v = 0 jusqu'à 1, pas de  (v/pas_v)
        début
            faire le produit [K] = [A] * [v3,v3,v,1]t
            pour u = 0 jusqu'à 1, pas de  (u/pas_u)
            début
                faire le produit   [u3,u2,u,1] * [K]
                Projeter le résultat 3D en 2D
                Tracer
            fin
        fin
    fin
```

Il est à noter que le temps de tracé des 32 patchs peut varier de façon importante d'une machine à une autre. Cela dépend évidemment de la configuration de chacun. Pour les plus chanceux, c'est à dire ceux équipés d'un 386 avec coprocesseur, l'attrait d'un tel programme est indéniable.

L'objet ainsi dessiné, il est aisé de le faire tourner et bouger dans l'espace par simulation de certaines touches du clavier. Les variations de quelques paramètres, dans la fonction de projection, suffisent à réaliser l'opération.

La programmation

Le programme teapot.c réalise donc le tracé, la projection, la translation et la rotation dans l'espace de notre théière. Après l'initialisation des paramètres de translation et de rotation, et celle, somme toute classique, du mode graphique, facilement personnalisable en modifiant la variable globale « type de carte » dans le fichier d'en tête teapot.h , on trace l'ensemble des points de l'objet par la fonction Points_Controle().



Points de contrôle de la théière

C'est avec cette modélisation fil de fer très simpliste que nous pourrons faire déplacer la théière, car le nombre de polygones à tracer est relativement réduit. La fonction Rotation_teapot() gère l'affichage et les mouvements de l'objet par appui sur des touches du clavier. Les touches flèches haute et basse traitent la rotation autour de l'axe des Y, tandis que celles de droite et de gauche permettent une rotation combinée en X et Z.

Les touches de fonction F1 ... F4 autorisent le déplacement en X et en Y, en avant et en arrière de l'objet, ce qui est utile quand l'objet n'est pas entièrement apparent ou centré.

Un dernier artifice des touches + et - opère sur la distance de l'oeil à l'objet créant ainsi un agrandissement ou un rétrécissement. La fonction Read_Points() récupère pour chaque patch les 16 points de contrôle, en ayant soin de décaler à chaque itération les points à lire, et c'est la fonction Patch_Bezier() qui calcule et trace le résultat final, c'est à dire la théière dans l'espace.

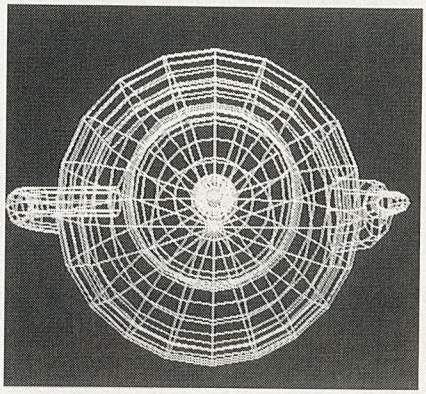
Une attention particulière a été apportée dans le calcul des polynomes en utilisant le schéma de Hörner, qui diminue le nombre d'opérations, gagnant ainsi en rapidité de calcul et ipso facto dans l'affichage (Horner_U(), Horner_V() et Get_Vecteur()).

Signalons ici que la fonction de projection Transform_3d_2d(), désormais classique, utilise la perspective pour projeter l'objet. Il est tout à fait possible d'y ajouter d'autres types de projection comme la parallèle par exemple. Les fonctions d'affichage C_X() et C_Y() permettent de dessiner la théière au centre de l'écran. Les paramètres trans_x et trans_y, une fois modifiés, contribueront à déplacer l'objet dans ce même repère.

Conclusion

Le programme reste largement ouvert à d'autres types d'expériences. Il constitue le premier pas vers une modélisation à facettes planes ou triangulaires (Cf [1]) pour permettre de tester des algorithmes de rendu (phong, gouraug), mais aussi de mapping de textures.

Mais avant d'en arriver là, bien des tests, et certainement des plus intéressants, restent à faire. D'abord, on peut modifier aisément les coefficients de la matrice [Mb], sans modifier le programme, et travailler ainsi dans une autre base telle que celle des B_splines des Beta-splines ou des cardinal splines, et obtenir ainsi un objet très malléable et déformable (Cf [3]) par modification de quelques paramètres.



Théière vue de dessus

D'autre part, on peut compléter sa collection d'objets à partir de la théière en ne conservant que le corps et la poignée, modélisant ainsi une tasse. On constate donc que bien des choses sont possibles à partir de cet objet, et que les esprits les plus inventifs ont permi et permettrons encore de lui appliquer les dernières techniques de l'informatique graphique.

Bibliographie

[1] Foley, Van Dam, Feiner, Hughes (1990). Computer Graphics, principles and practice. Addison-Wesley.

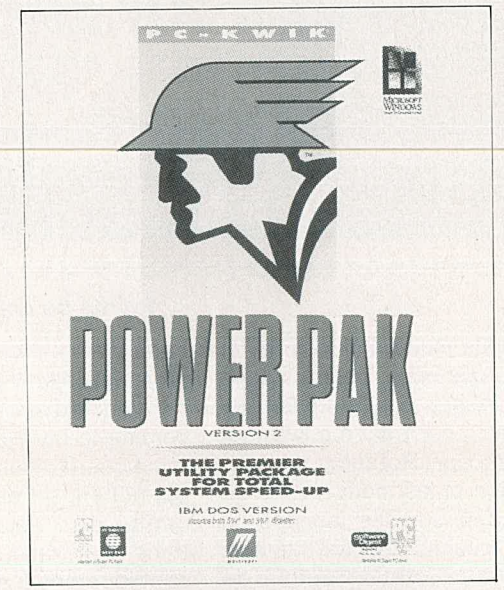
[2] Frank Crow (1987). The origins of the teapot. IEEE Computer graphics and applications january 1987.

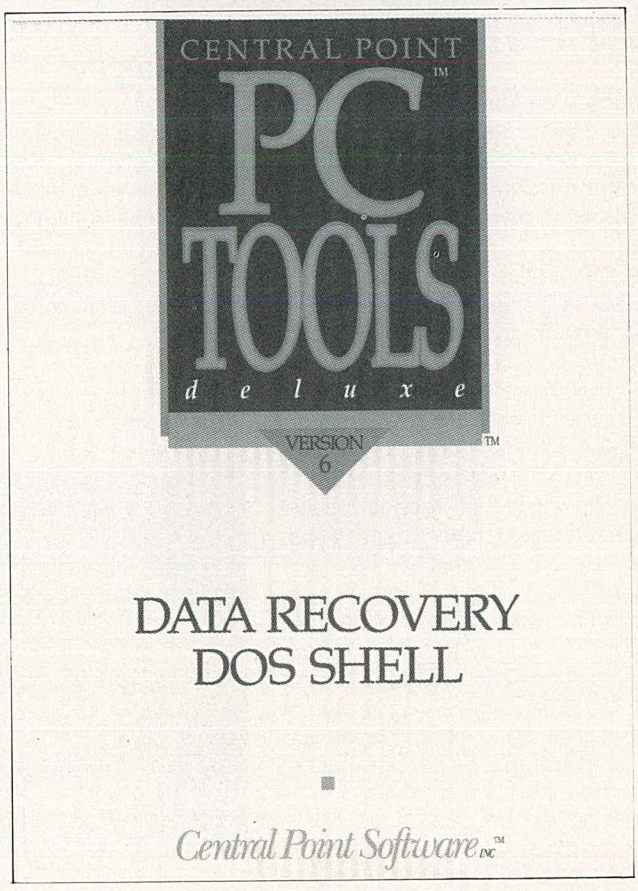
[3] Alan Watt (1989). Fundamentals of Three-Dimensional ComputerGraphics. Addison-Wesley.

Nous remercions M. André Stanek pour sa collaboration photographique.

ACCELEREZ VOTRE PC

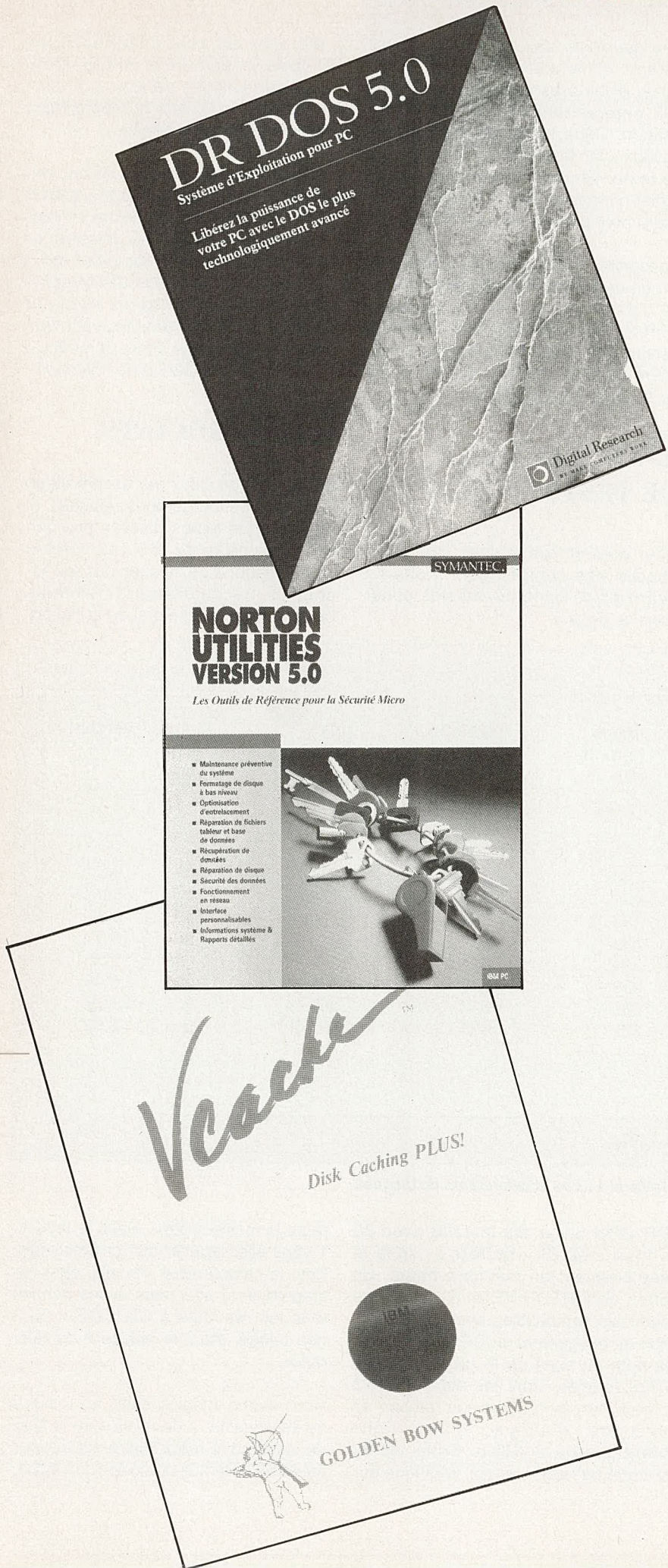
par ***G. Korwin***





Peu importe le processeur installé dans votre ordinateur, vous atteindrez rapidement ses limites. La plupart du temps ce n'est pas la vitesse du processeur qui définit les performances du système, c'est le temps de réponse des disques durs.

[SERVICE LECTEURS : INTERESSANT : 25 / MOYEN : 26 / MAUVAIS : 27]



Les limites de performances

Pour accéder aux données, même un i486 le plus rapide sera obligé d'attendre pendant l'équivalent de dizaines de milliers d'instructions. Le processeur et la mémoire centrale travaillent très rapidement - ce sont les dispositifs électroniques. Un disque dur, surtout s'il communique avec le processeur à travers un bus 8 bits, n'est qu'un dispositif mécanique, incomparablement plus lent que tout dispositif électronique.

Evidemment, pour travailler plus vite « on n'a qu'à » changer la machine. Installer un disque rapide et un contrôleur caché. Bien qu'on puisse trouver actuellement des machines performantes à un prix abordable, cette solution peut coûter quelque dizaines de milliers de francs.

Une solution économique et relativement efficace est d'installer un logiciel d'accélération de disques.

Solution logicielle

Les logiciels d'antémémoire (parfois appelés « disques cache ») recopient les données le plus souvent utilisées dans la mémoire centrale. Chaque fois qu'une application accède aux fichiers, le logiciel cache vérifie si les données requises se trouvent déjà dans un tampon en mémoire centrale et, dans l'affirmative, les utilise au lieu de les chercher sur le disque dur. Cette opération évite le passage systématique par la mécanique du disque dur et accélère chaque opération de manipulation de données.

En pratique, si votre travail nécessite des accès répétitifs aux mêmes données (d'un tableur ou d'une base de données comptable, par exemple), les logiciels d'antémémoire permettent d'accélérer votre système, en contrepartie d'un coût relativement faible.

Une réalisation pratique d'un disque cache a été décrite dans notre numéro 12. Aujourd'hui, nous vous présentons une évaluation comparative de six logiciels de commerce.

■

LES CANDIDATS

De plus en plus d'ensembles d'utilitaires sont vendus avec un logiciel cache inclus. Parmi ceux qui sont présents sur le marché, nous avons choisi DR Cache livré en standard avec DR DOS 5.0 de Digital Research, PC Cache inclus dans l'ensemble PC Tools version 6, N Cache livré avec les Outils NORTON version 5.0, PC Kwik livré dans l'ensemble POWERPAK de Multisoft et Smartdrive livré avec WINDOWS 3.0 de Microsoft. L'outsider est vendu tout seul : V Cache de Golden Bow Systems.

BANC DE TEST

Notre ensemble de test se veut être au plus près de la situation réelle d'une utilisation quotidienne. Pour cela, nous avons crée un programme de test qui manipule extensivement les fichiers sous FoxPRO version 1.0.

Un premier fichier de 20,857 Mo comporte 38.558 enregistrements de 541 octets chacun. A quatre reprises, le logiciel de test procédait au remplacement du contenu de rubriques, reindexation dans l'ordre composé de plusieures rubriques et la recopie physique du fichier selon ce nouveau index. Cette tâche correspond à des longs travaux sur des bases de données comptables ou sur les grands fichiers d'adresses.

Une deuxième série de test consistait en une même manipulation d'un petit fichier d'adresses de 512Ko, divisé en 997 enregistrements de 512 octets chacun. Cette tache correspond aux travaux avec les tableurs ou traitement de texte. En plus, la structure de ce fichier correspond à la taille de tampon choisi pour chacun des accélérateurs.

L'ensemble de programmes s'exécutait sur un ordinateur ZENITH 386/16 équipé d'un disque dur de 119 Mo MAXTOR de 28ms d'accès, sous DR DOS version 5.0. Cet ordinateur de « première génération 386 » correspond à un AT basé sur 286 rapide.

DEROULEMENT DE TESTS

Pour garantir les mêmes chances à chacun des compétiteurs, nous les avons placé dans les mêmes conditions de départ.

DR DOS 5.0 a été installé avec 30 « files » et 15 « buffers », sans le déplacement en mémoire haute. En effet, SMARTDRIVE et V Cache ne sont pas compatibles avec cette fonction de relogement du DOS (voir aussi la note au sujet de la rapidité du DR DOS relogé). Tous les accélérateurs ont été forcés à utiliser un tampon en mémoire étendue, d'une taille identique de 2048Ko. Enfin, avant chaque passage de test, le disque dur a été optimisé avec COMPRESS de PC Tools version 6.0 et SPEED DISK de Norton Utilities version 4.5, pour placer tous les fichiers et répertoires à la même position de départ.

SPEED DISK de Norton Utilities version 5.0 n' effectuait pas correctement l'optimisation, s'arrêtait en cours sans aucune raison apparente, laissant les fichiers fragmentés et éparpillés sur la surface du disque. Selon SYMANTEC, la version 5 des Utilitaires n'est pas adaptée à DOS 5, une nouvelle version sera annoncée concurremment à l'annonce de MS DOS 5 par Microsoft.

LES RESULTATS

En conclusion, tous les accélérateurs testés ont donné de bons résultats, en réduisant les temps d'accès aux données de l'ordre de 30%. Le meilleur produit dépend de ce que vous recherchez : l'accélération maximale, l'encombrement minimal de la mémoire ou le meilleur prix... Dans le tableau 1 vous allez trouver des chiffres réels. Pour la comparaison visuelle, dans les graphiques 2 et 3 nous avons ramené tous les résultats à DR DOS « nu » non relogé, étant le niveau 0 de référence.

Nous avons imaginé aussi un « indice de satisfaction »: de l'indice de la taille de la memoire restant libre nous avons soustrait le temps d'accès et un indice

	temps f petit sec	temps f grand min	memoire libre ko	prix F. TTC	langue	version
DR DOS déplacé	404	770	625	1,174.14	fr	5.0
DR DOS nu	252	431	566	1,174.14	fr	5.0
DR CACHE	194	385	554	1,174.14	fr	0.06
PC CACHE	182	332	537	1,529.95	fr	6.0
N CACHE	85	324	491	1,719.70	fr	5.00
V CACHE	123	359	549	1,174.14	us	5.06
SMARTDRIVE	185	385	547	2,360.14	fr	3.03
PC KWIK	194	382	503	1,897.60	us	2.02

Tableau 1 : *Les accélérateurs de disques*

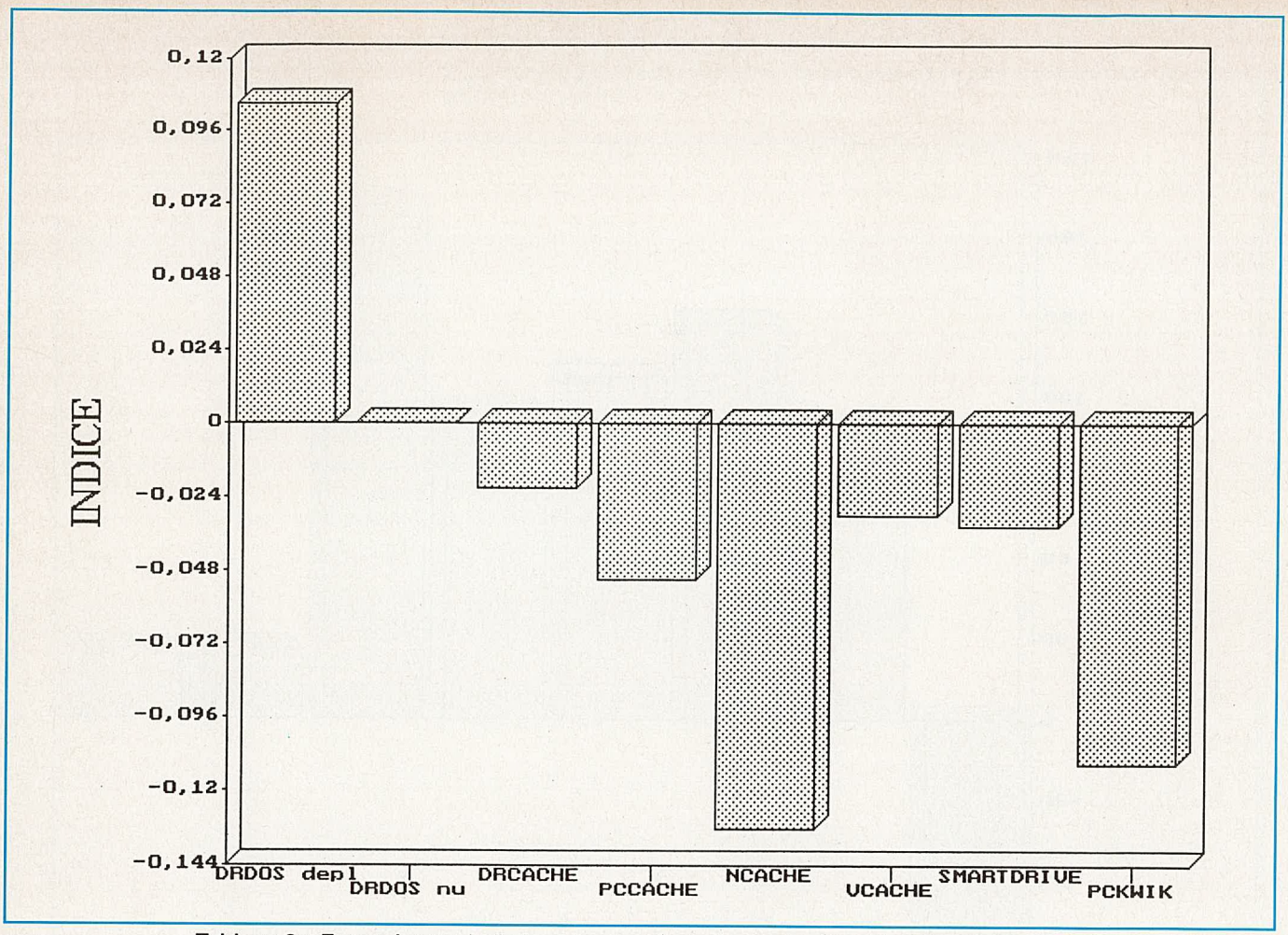


Tableau 2 : *Encombrement de la mémoire centrale par rapport à DR DOS « nu ».*
Indice positif indique la meilleure performance.

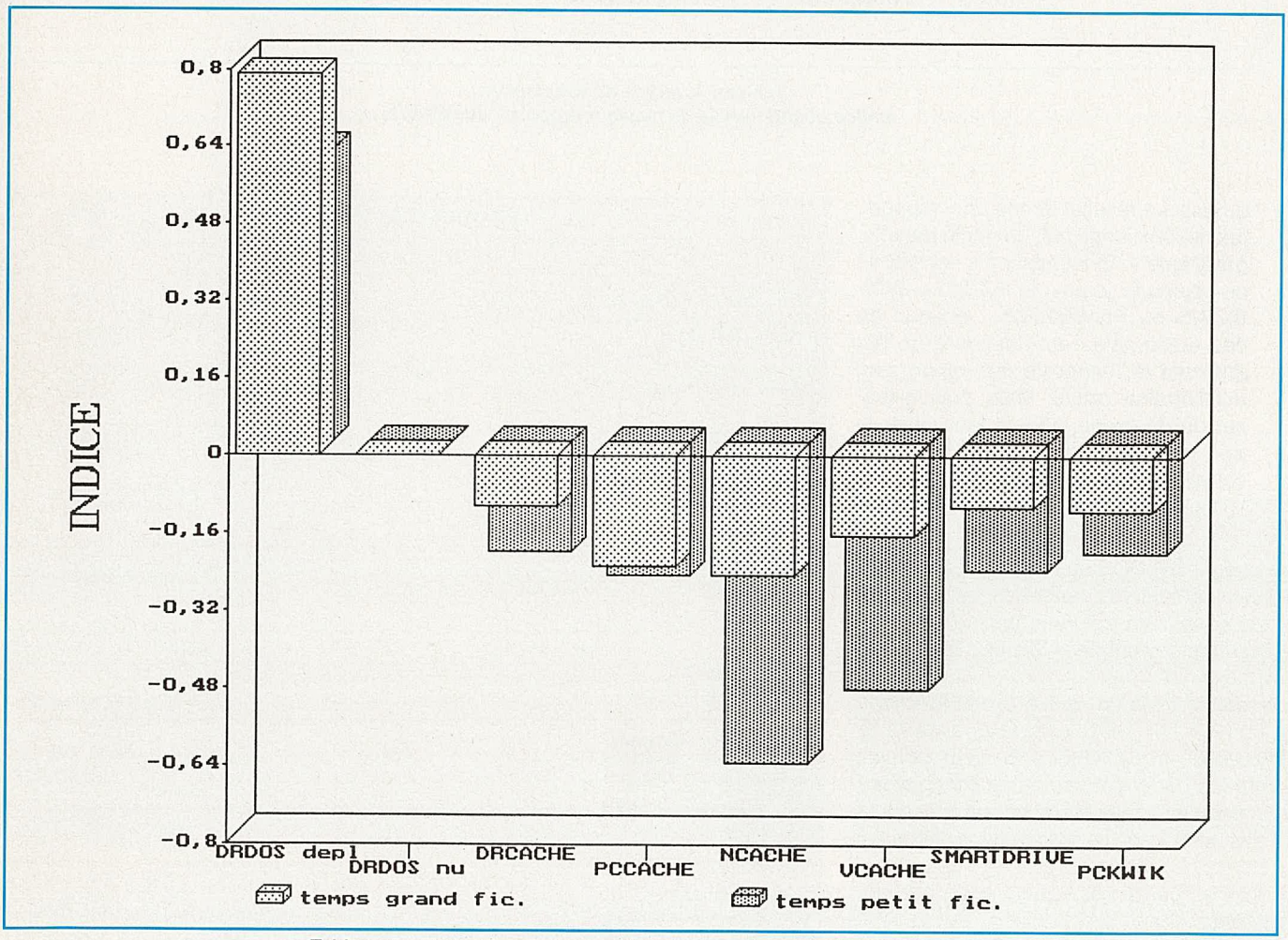


Tableau 3 : *Réduction de temps d'accès par rapport à DR DOS « nu ».*
Indice négatif indique la meilleure performance.

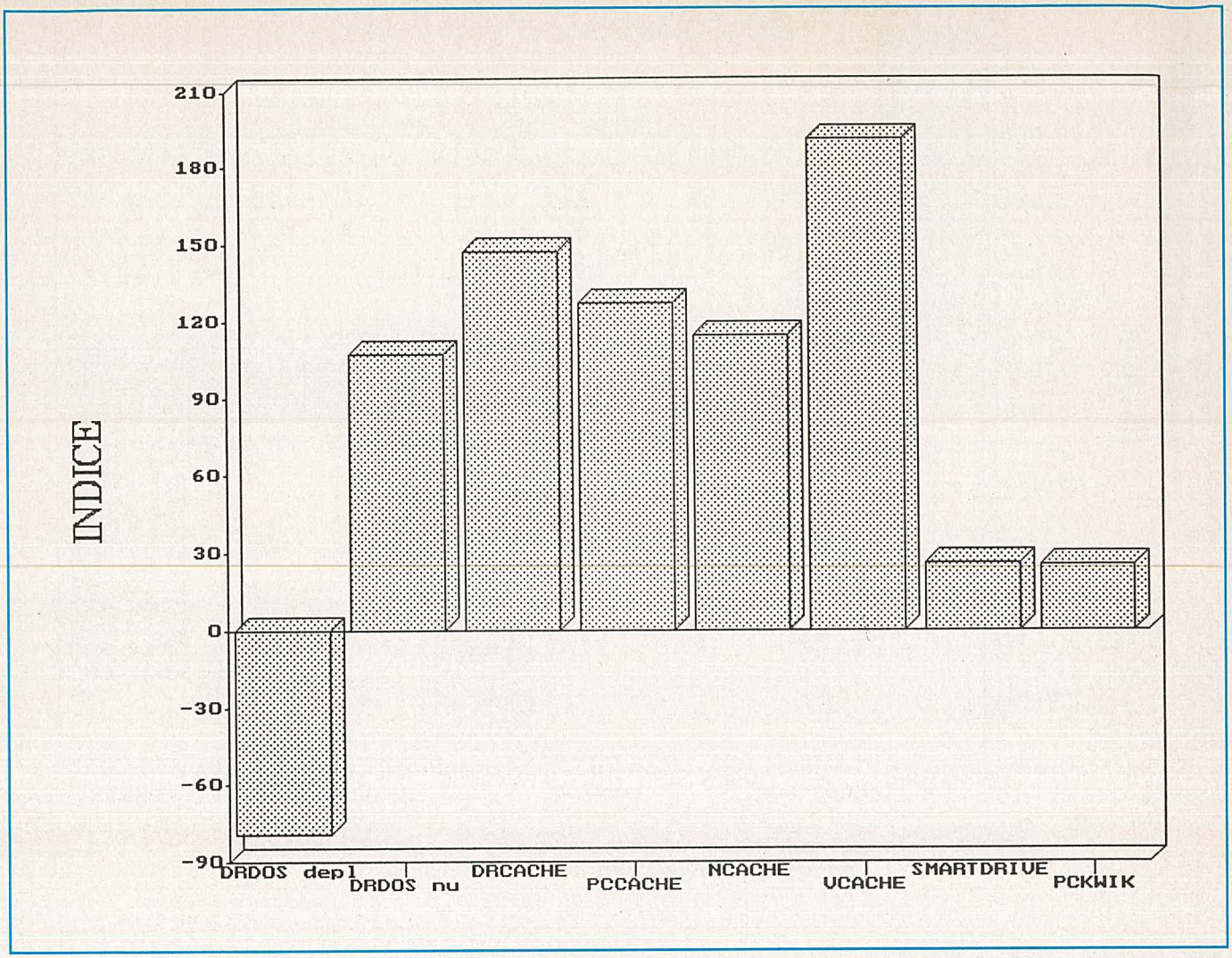


Tableau 4: ***Indice de satisfaction.***
Indice positif indique le meilleur rapport « qualité/prix »

du prix. Le résultat donne un « rapport qualité/prix » optimal, présenté dans le graphique 4. Evidemment, il est difficile de comparer le prix de PC Tools, WINDOWS ou POWERPAK : chacun de ces en-sembles est destiné à un but différent et n'apporte pas qu'un seul accélérateur cache. Mais, pour disposer du dit cache, il vous faut payer le prix indiqué, dans cette optique une certaine comparaison est possible...

NOTE DR DOS

Le fait de reloger DR DOS en mémoire haute fait dégrader notablement ses performances.
Le temps d'exécution est rallongé de quelque 50% - le système devient deux fois plus lent.

Ainsi, lors de l'installation, grâce à des fonctions interactives de DR DOS (questions dans le AUTOEXEC.BAT et CONFIG.SYS) vous pouvez choisir entre les performances et les économies : soit vous choisissez la taille maximale de la mémoire disponible, soit la rapidité du système.

Ce phénomène nous a été signalé aussi par plusieurs lecteurs.

PRODUITS TESTES

DR Cache version 0.06
livré avec DR DOS 5.0 en français
inclus dans le prix de 1 174.14 Frs TTC
DIGITAL RESEARCH
BP 303 - 41 bis av de l'Europe
78144 Vélizy CEDEX
téléphone : (1) 39 46 33 66

N Cache version 5.00
livre avec Norton Utilities 5.0 en français
inclus dans le prix de 1719.70 Frs TTC
édité par **SYMANTEC**
31-35 rue Gambetta
92150 Suresnes
téléphone : (1) 47 72 16 50
distribué par **FRAME**
32 bis rue V Hugo
92800 Puteaux
téléphone : (1) 47 72 77 77

V Cache version 5.06
en anglais (incompatible avec
le relogement du DOS)
prix 1 174.14 Frs TTC
édité par **Golden Bow Software**
2665 Ariane Drive Suite 207
San Diego, CA 92117 USA
téléphone : 19 (1) 483 0901
importé par **CFAO SYSTEME**
34 rue Pergolèse
75116 Paris
téléphone : (1) 40 67 99 90

PC Cache version 6.0
livré avec PC Tools 6.0 en français
inclus dans le prix de 1 529.95 Frs TTC
édité par **Central Point Software**
15220 NW Greenbrier Pkwy suite 200
Beaverton, OR 97006 USA
téléphone : 19 (1) 503 690 8090
distribué par **SOFTPUBLISHING**
124 bd de Verdun
92411 Courbevoie CEDEX
téléphone : (1) 46 67 22 44
et **LOGONE**
36 Cours du Chapeau Rouge
33000 BORDEAUX
Tél : 56 44 21 11

Smartdrive version 3.03
(nécessite HIMEM 2.60,
incompatible avec le relogement du DOS)
livré avec WINDOWS 3.0 en français
inclus dans le prix de 2 360.14 Frs TTC
édité par **MICROSOFT**
12 av du Quebec
ZA Courtaboeuf
91957 les Ulis CEDEX
téléphone : (1) 69 86 46 46

PC KWIK version 2.02
livré dans l'ensemble POWERPAK en anglais
prix 1897.60 Frs TTC
édité par **Multisoft**
15100 SW Koll Parkway
Beaverton, OR 97006, USA
téléphone : 19 1 (503)
distribué par **SOFTMART**
23 rue de Clichy
BP 130
93403 St Ouen CEDEX
téléphone : (1) 40 10 25 25
(version francaise annoncée courant juillet)

Les menus déroulants sont des éléments indissociables de l'interface utilisateur de Windows. Grâce à eux, les applications Windows disposent d'un jeu de commandes intuitif et, au moins en partie, standardisé. Une application dispose, en général, au moins des menus « Fichier » et « Edition », qui varient peu d'un programme à un autre. Le rôle de ces menus est parfaitement défini et l'utilisateur n'a aucun mal à s'y retrouver. Créer des menus dans une application est tout ce qu'il y a de plus facile; la tâche la plus compliquée est de définir les libellés des menus de sorte à ce qu'ils soient aussi intuitifs que possible. Ce sont donc les multiples façons de créer des menus que nous allons maintenant étudier.

Un peu d'anatomie.

Les éléments d'un menu peuvent être regroupés en deux structures. La première, qui existe toujours, est la barre de menus (ou menu principal, ou encore « top menu »), qui est affichée juste en dessous de la barre titre d'une fenêtre. Ce menu principal contient un certain nombre d'éléments qui, le plus souvent, donnent accès à une deuxième structure qui est dite « menu secondaire » (ou « pop-up menu »). Contrairement à certaines autres interfaces utilisateur qui font appel aux fenêtres et aux menus, Windows ne permet pas d'autres niveaux de sous-menus (tout du moins, pas dans la version 2.11) : à l'instar du Macintosh les éléments des sous-menus donnent accès à des boites de dialogue si des données complémentaires sont nécessaires pour l'exécution de la fonction correspondante.

Cette notion n'est plus exacte dans Windows 3.0, qui permet d'associer un menu « descendant » à n'importe quel élément d'un menu. Les menus « pop-up » peuvent maintenant s'afficher n'importe où à l'écran. C'est particulièrement important lorsque l'application s' exécute sur un écran à haute résolution: on n'a plus à promener la souris sur des kilomètres de table pour appeler une fonction dans le menu.

Un troisième type de menu, créé automatiquement par Windows, peut être associé à tout type de fenêtre (y compris les fenêtres enfants). Il s'agit du menu système, qui est représenté par une petite case à la gauche de la barre de titre. Ce menu contient un certain nombre de fonction prédéfinies qui permettent de jouer sur l'état de la fenêtre: taille, position, activation, fermeture, etc. Vous l'avez certainement remarqué, les libellés des menus sont fréquemment accompagnés de noms de touches: Alt-F4, Shift-Ins, Del etc. Ces touches, ou ces combinaisons de touches, donnent accès aux mêmes fonctions que les éléments correspondants des menus. Elles évitent cependant d'avoir à parcourir toute l'arborescence d'un menu pour exécuter une commande. Nous verrons en détail plus loin comment ajouter à vos menus ces accélérateurs.

A partir de la version 2 de Windows, une importante modification a été introduite. Il s'agit de la possibilité d'activer une commande dans un menu à l'aide d'une lettre de son nom. Lors de l'affichage du menu, cette lettre sera soulignée. Windows reconnaît ces lettres, soit en combinaison avec la touche Alt dans le menu principal, soit isolément dans les sous-menus.

Windows offre deux façons distinctes de créer un menu. La première (et la plus facile) consiste à définir un menu en tant que ressource : le menu est décrit dans le fichier .RC et l'application n'a plus qu'à le charger au moment de l'exécution. L'autre façon, un peu plus compliquée, consiste à faire appel aux fonctions de construction des menus pour créer dynamiquement un menu.

Ces deux moyens ne sont nullement exclusifs. Un menu défini dans un fichier ressources peut être à tout moment modifié par l'application pour refléter un changement de l'environnement. Ajout ou inactivation des éléments ne sont qu'une question d'un appel à la fonction correspondante. La nature du menu ne constitue pas une limitation: il est possible d'ajouter des éléments même au menu système. C'est ainsi que l'on retrouve fréquemment dans le menu système de la fenêtre principale d'une application la commande « A propos de...».

Le menu ressource

Le mini-langage du compilateur de ressources (RC.EXE) contient un certain nombre de mots-clés qui permettent de définir rapidement la structure d'un menu. Voici, par exemple, la définition d'un petit menu:

```
#define DEMO_OPEN     100
#define DEMO_CLOSE    101
#define DEMO_QUIT     102
Demo MENU
 BEGIN
  POPUP « &Fichier »
   BEGIN
    MENUITEM « &Ouvrir »,DEMO_OPEN
    MENUITEM « &Fermer »,DEMO_CLOSE
   END
  MENUITEM « &Quitter! »,DEMO_QUIT
 END
```

Cette structure crée, après compilation, un menu identifié par le nom « Demo ». L'affichage de « Demo » aboutit à une barre de menu qui contient les libellés « Fichier » et « Quitter! ». La fonction « Fichier » (POPUP) donne accès à un sous-menu qui contient les libellés «Ouvrir » et « Fermer ». Notez que les libellés des éléments du menu principal qui ne donnent pas accès à un sous-menu sont, par convention, suivis d'un point d'exclamation. Examinons maintenant un peu plus en détail ce petit exemple.

MENU est le mot-clé qui marque le début de la définition d'un menu. Il est précédé d'une chaîne de caractères qui identifie de façon unique la ressource (ici, il s'agit de « Demo »). Notez au passage qu'il n'y a aucun obstacle à ce que le même nom soit affecté à plusieurs ressources, du moment qu'il s'agit de ressources différentes (par exemple, un menu et une icône). MENU peut être en option suivi de deux groupes de mots-clés qui définissent quand et comment le menu sera chargé en mémoire. Par défaut, ces options sont LOADONCALL et MOVEABLE/DISCARDABLE. Cela signifie que le menu ne sera chargé à partir du fichier ressources que sur demande et que la zone mémoire qu'il occupe est mobile et libérable en cas de besoin.

Le fait qu'un menu soit DISCARDABLE a une incidence pratique si vous modifiez dynamiquement ses constituants: si, pour une raison quelconque, Windows doit récupérer de la place mémoire et qu'il est amené à libérer la mémoire occupée par le menu, toutes les modifications seront perdues. Il vous appartiendra alors de reconstituer toutes les modifications lorsque le menu aura été de nouveau rappelé en mémoire.

Le couple BEGIN et END marque les limites d'une description de menu ou de sous-menu. Il est requis même s'il n'encadre qu'une seule déclaration. RC.EXE se plaint bruyamment si cette syntaxe n'est pas respectée.

POPUP marque le début de la descrip-

tion d'un sous-menu. Il est suivi par une chaîne de caractères qui deviendra, dans le menu principal, le libellé donnant accès au sous-menu. Ce mot-clé ne peut pas être employé à l'intérieur de la description d'un sous-menu.

MENUITEM sert à déclarer un élément de menu (principal ou pop-up). Il est suivi du libellé de l'élément ainsi que d'une étiquette qui l'identifie. Ces étiquettes (DEMO_OPEN, DEMO_CLOSE, etc.) doivent avoir été au préalable déclarées et initialisées à des valeurs qui permettront de les reconnaître sans ambiguïtés. Il est pratique de définir ces étiquettes dans le fichier .H de l'application et de l'inclure au début du fichier ressources (avec #include). De cette façon, vous pourrez référencer les étiquettes dans le code source directement par leur nom.

Puisque les ressources d'une applications sont incluses dans le fichier exécutable, tout ce qu'il y a à faire pour utiliser un menu ainsi défini consiste à a) charger (si nécessaire) la ressource correspondante et b) l'associer à une classe de fenêtres ou à une fenêtre particulière.

Le menu peut être utilisé en tant que menu par défaut pour toute une classe de fenêtres. Dans ce cas, il suffit d'initialiser l'élément correspondant de la structure de définition d'une classe sans qu'il soit nécessaire de charger le menu au préalable:

```
wndclass.lpszMenuName=(LPSTR)"Demo ";
```

Il peut être aussi utilisé pour une fenêtre unique. Il faut dans ce cas charger spécifiquement la ressource correspondante. Le chargement se borne à initialiser un vecteur (hMenu dans l'exemple) de type HMENU puis à utiliser ce vecteur en tant que neuvième paramètre de la fonction CreateWindow() (les autres paramètres sont volontairement omis):

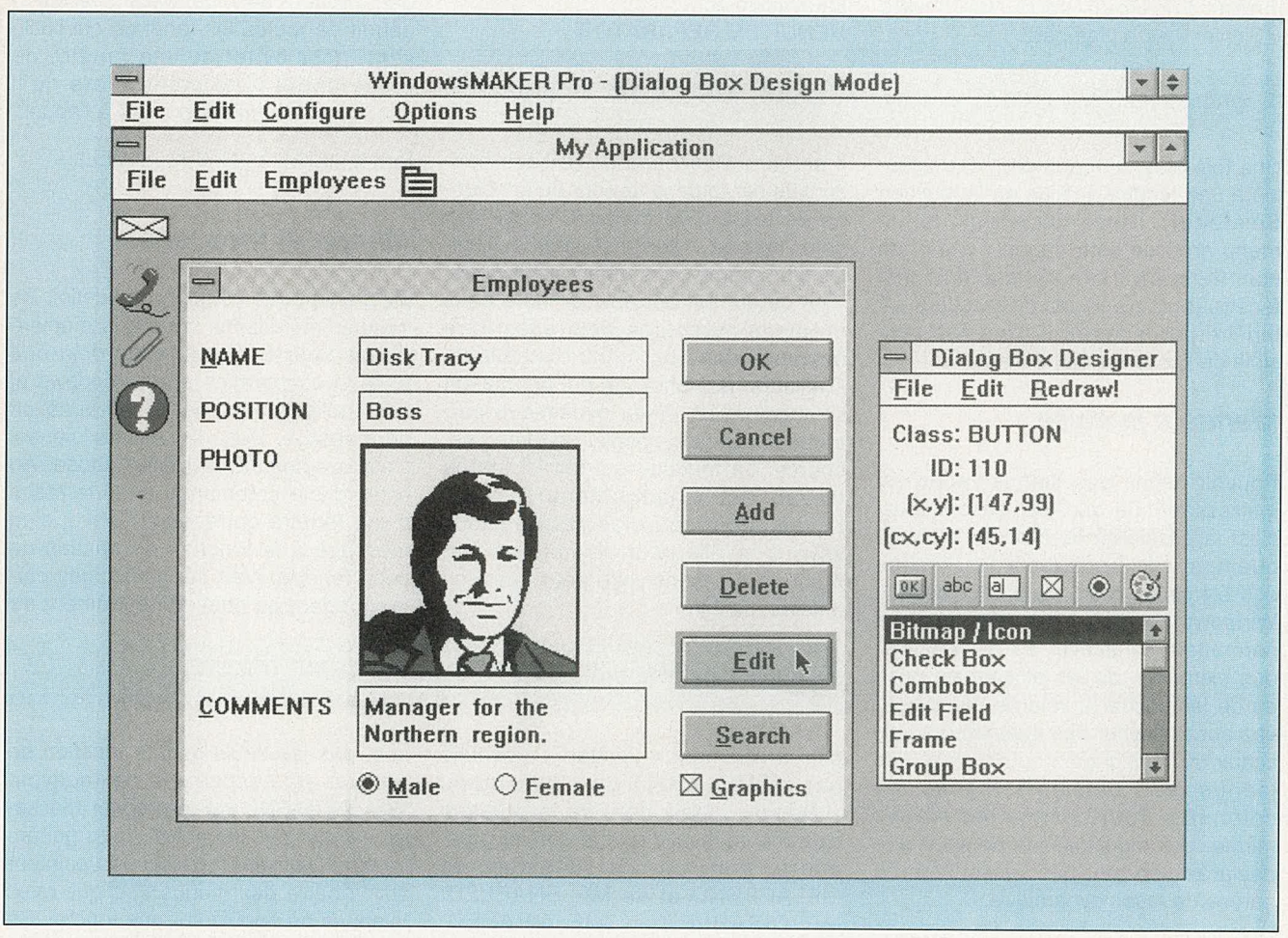
```
hMenu=LoadMenu(hInstance,"Demo");
hWnd=CreateWindow("NomClasse","Titre",,,,,,,hMenu,,);
```

Dans ce cas, le menu n'apparaîtra que dans la fenêtre qui vient d'être créée. Toutes les autres fenêtres afficheront le menu défini lors de la création de la classe correspondante (ou rien si aucun menu n'a été associé).

Enfin, un menu peut être associé à une fenêtre après sa création, à l'aide de la fonction SetMenu():
SetMenu(hWnd,hMenu);
Cette dernière forme permet de changer dynamiquement les menus d'une fenêtre.

Le menu dynamique

La définition d'un menu dans un fichier ressources est la manière la plus aisée d'incorporer un menu dans une fenêtre. Mais ce n'est pas forcément la technique la plus rentable. Imaginez que vous ayez, dans votre application, une ou plusieurs fenêtres dont les menus doivent subir, au cours de l'exécution, des changements profonds pour refléter les modifications de l'environnement. Vous pourriez définir, malgré tout, les menus par défaut dans le fichier .RC, mais cela suppose un pari sur la configuration de départ de l'ap-



Vos programmes, simples ou complexes, sont maintenant limités seulement par votre imagination.

plication. Si vous perdez votre pari, tout le menu est à reconfigurer. Il est plus intéressant dans ce cas de ne pas créer de menu par défaut, mais d'apprécier au préalable la situation et de construire, en fonction de l'environnement, un menu sur mesure.

Pour vous aider dans cette démarche, Windows permet de créer un menu dans sa totalité en utilisant seulement deux fonctions appelées CreateMenu() et ChangeMenu().

La fonction CreateMenu() ne fait qu'initialiser un menu vide; elle retourne un vecteur de type HMENU sur ce menu: hMenu=CreateMenu();

Le rôle de ChangeMenu() est d'insérer des libellés dans ce menu:

```
ChangeMenu(hMenu,NULL,"&Quitter!",DEMO_QUIT,MF_APPEND);
```

La difficulté réside dans le fait que les sous-menus doivent être créés séparément et insérés ensuite dans le menu principal. En supposant que hSub soit un vecteur sur un sous-menu contenant les libellés « Ouvrir » et « Fermer », l'ajout de ce sous-menu au menu principal sous le libellé « Fichier » serait fait par:

```
ChangeMenu(hMenu,NULL,"&Fichier",hSub,MF_APPEND|MF_POPUP);
```

Une fois crée, le menu peut être associé à une fenêtre à l'aide de la fonction SetMenu(). Notez cependant qu'un menu crée de cette façon n'a pas de nom. Il ne peut être assigné qu'à une fenêtre (ou à plusieurs, en le rattachant fenêtre par fenêtre). Il n'est pas possible de l'associer à une classe.

Menus à la carte

Pouvoir définir des menus variés est intéressant. Ce qui l'est encore plus, c'est la possibilité de modifier dynamiquement non seulement les libellés des éléments, mais aussi leur aspect. Windows comporte des fonctions qui permettent d'inactiver ou de réactiver des éléments, de les afficher en grisé ou de les cocher à volonté. Vous pouvez aussi afficher des séparateurs verticaux ou horizontaux ou de créer des menus sur plusieurs lignes ou colonnes. Tout comme les menus même, ces modifications peuvent être déclarées en tant que ressources ou gérées de façon dynamique.

Examinons tout d'abord de quelle manière nous pouvons modifier l'aspect des menus créés en tant que ressources. Les éléments des menus sont déclarés à l'aide du mot-clé MENUITEM. Ce que je ne vous ai pas dit plus haut est que le dernier élément de cette déclaration, à savoir l'étiquette, peut être suivi d'un qualificateur qui détermine comment le libellé sera affiché. Voici la liste de ces qualificateurs:

CHECKED: le libellé comporte une marque (est « coché ») à gauche;
GRAYED: le libellé est inactif et ne peut être choisi. Le texte est affiché en gris;
INACTIVE: comme **GRAYED,** mais le texte reste affiché en noir;
MENUBREAK: Le libellé et tous les libellés suivants apparaissent dans une nouvelle colonne;
MENUBARBREAK: comme **MENUBREAK**, mais avec une barre verticale pour séparer les deux colonnes.
GRAYED et **INACTIVE** ou **MENUBREAK** et **MENUBARBREAK** ne peuvent pas être utilisés ensemble. Il existe encore un mot-clé, **SEPARATOR.** Sa syntaxe est:
MENUITEM SEPARATOR

Il entraîne l'affichage d'une ligne horizontale sur toute la largeur de la colonne de texte d'un menu. Comme son nom l'indique, il sert à séparer les libellés d'un même sous-menu en groupes. Tout ce qui est possible de faire dans un menu ressource peut aussi être accompli dans un menu dynamique. D'ailleurs, les fonctions qui permettent de modifier un menu dynamique permettent aussi de modifier un menu ressource. La fonction qui fait ici tout le travail est ChangeMenu(). Nous l'avons déjà vue plus haut et nous savons qu'elle peut ajouter des éléments à un menu; elle peut aussi en enlever; la ligne :

```
ChangeMenu(hMenu,DEMO_EXIT,NULL,NULL,MF_DELETE);
```

détruit l'élément « Quitter! » (identifié par DEMO_EXIT) de notre menu « Demo ». Mais, lors de la création d'un menu, il suffit que le dernier paramètre soit un OU logique de MF_APPEND et de MF_ENABLED, MF_DISABLED, MF_GRAYED, MF_UNCHECKED, MF_CHECKED, MF_MENUBREAK ou MF_MENUBAR BREAK pour que l'élément ajouté ait l'aspect désiré. Si vous omettez MF_APPEND et que le deuxième paramètre de ChangeMenu() soit l'étiquette d'un élément déjà ajouté, vous pouvez changer les libellés existants. Notez au passage qu'il existe une fonction qui permet de modifier rapidement l'état coché ou non d'un élément. Il s'agit de la fonction CheckMenuItem(). La ligne:

```
CheckMenuItem(hMenu,DEMO_OPEN,MF_CHECKED);
```

coche l'élément du menu hMenu désigné par DEMO_OPEN. S'il s'agit de savoir quels sont les attributs d'un élément, un appel à:

```
wFlags=GetMenuState(hMenu,DEMO_OPEN,MF_BYCOMMAND);
```

retourne une valeur qui est une combinaison logique des attributs possibles. Si vous avez une indigestion des menus, sachez que vous pouvez vous en débarrasser grâce à la fonction DestroyMenu(). Elle a pour paramètre le vecteur sur un menu. Il est cependant prudent de « détacher » le menu détruit de toutes les fenêtres qui l'utilisent, pour éviter qu'une fenêtre ne fasse appel à ce menu alors qu'il n'existe plus. Utilisez pour cela l'appel:

SetMenu(hWnd,(HMENU)NULL);

Menus et messages

Maintenant que nous savons définir les menus, intéressons nous à la manière de les utiliser. Vous savez déjà que toute la coopération entre Windows et une application est basée sur la notion de message. Dans le cas des menus, c'est exactement la même chose. Au choix d'une commande dans le menu d'une fenêtre correspond l'envoi d'un message à la fonction de gestion de cette fenêtre. Mais les messages correspondant au choix d'un élément ne sont pas les seuls qu'une fonction de gestion est susceptible de recevoir. WM_INITMENU est le premier message envoyé par Windows. Il est généré lorsque l'utilisateur déplace le pointeur sur la barre des menus et clique avec la souris ou appuie sur une touche. Ce message est posté juste avant que Windows n'affiche le menu et vous

donne la possibilité de changer l'état des libellés avant qu'ils ne soient affichés. Windows envoie un message semblable, WM_INITMENUPOPUP, juste avant qu'un sous-menu ne soit affiché. Dans les deux cas, le paramètre wParam contient le vecteur sur le menu ou le sous-menu. Nous avons vu plus haut que, si vous modifiez l'aspect d'un menu défini en tant que ressource et que Windows est obligé de libérer la place occupée par ce menu en mémoire, vos modifications sont perdues. Si vous avez songé à conserver une copie de ces modifications dans des variables, c'est au moment où votre application reçoit le message WM_INITMENU qu'il faut en profiter pour les répercuter sur le menu qui vient d'être rechargé en mémoire. Au fur et à mesure que l'utilisateur déplace la barre de sélection (que ce soit avec la souris ou avec le clavier), Windows poste une série de messages WM_MENUSELECT. Le paramètre wParam contient l'étiquette de l'élément actuellement choisi. Le mot de poids faible de lParam contient l'aspect du libellé (coché, gris, inactif etc) et le mot de poids fort est le vecteur sur le menu. Ces messages permettent, par exemple, d'afficher quelque part dans la fenêtre de l'application, un petit texte d'aide pour chacune des commandes du menu.

Ces trois message peuvent être, dans la majorité des cas, ignorés par l'application et passés à la fonction DefWindowProc(). Le seul message qui doit mériter une attention toute particulière est WM_COMMAND: il indique que l'utilisateur a choisi un élément actif du menu. Le paramètre wParam contient l'étiquette de l'élément choisi, par exemple DEMO_QUIT si l'utilisateur a choisi l'élément « Quitter! » de notre petit menu.

En ce qui concerne le menu système, la seule différence est que le message reçu par l'application en cas de choix d'un élément est WM_SYSCOMMAND au lieu de WM_COMMAND. Les valeurs possibles de wParam sont décrites dans les pages 596-597 du manuel « Programmer's reference » du kit de développement et correspondent aux commandes qui se trouvent habituellement dans ce menu. A ces valeurs viennent s'ajouter les étiquettes des éléments que vous auriez pu ajouter au menu système dans le cadre de votre application. Il est important que les messages WM_SYSCOMMAND soient passés à DefWindow Proc() sinon les commandes du menu système ne sont pas traitées.

Accélérateurs, ou comment être servi plus vite

Les accélérateurs sont des séquences de touches qui aboutissent à la génération par Windows de messages WM_COMMAND ou WM_SYSCOMMAND. Leur rôle est de reproduire à partir du clavier les actions des commandes les plus utilisées. Toutes les opérations, depuis l'interception des messages WM_KEYDOWN et WM_KEYUP jusqu'à l'envoi de WM_COMMAND ou WM_SYSCOMMAND sont prises en charge par Windows, qui n'a besoin que d'une table qui décrit la correspondance entre les séquences de touches et les messages correspondants. Une telle table est définie en tant que ressource et en voici un exemple:

```
Demo ACCELERATORS
 BEGIN
  VK_F3,IDM_OPEN,VIRTKEY
  "^X",IDM_QUIT
  88,IDM_QUIT,ASCII,CONTROL
 END
```

Dans cet exemple, la touche F3 (VK_F3) entraîne l'envoi du message WM_COMMAND avec la valeur IDM_OPEN vers la fenêtre principale de votre application, alors que la séquence CtrlX (^X) entraîne l'envoi du même message avec la valeur IDM_QUIT (en supposant que IDM_ OPEN et IDM_QUIT aient été définies au préalable).

Remarquez que la deuxième et la troisième définition sont identiques (88 est le code ASCII du caractère 'X'), mais ^X spécifie directement un caractère accompagné de la touche Control, alors que dans la seconde définition, il est explicitement spécifié que 88 est un code ASCII et que la touche Control doit être pressée pour que la séquence soit reconnue. Il est possible de spécifier SHIFT à la place de CONTROL pour utiliser les touches majuscules (c'est valable aussi bien pour les caractères ASCII que pour les touches spéciales).

Les accélérateurs n'ont pas besoin d'options de chargement. Vous pouvez définir plusieurs tables si votre application en a besoin.

Une fois que votre table a été définie, il ne vous reste plus qu'à indiquer à Windows comment l'utiliser. Pour cela, il faut, dans WinMain, définir un vecteur sur la table et charger celle-ci:

```
HANDLE hAccel;
hAccel=LoadAccelerators(hInstance,"Demo");
```

Puis il faut modifier, toujours dans WinMain, la boucle qui reçoit les messages et les expédie aux fenêtres de votre application:

```
while(GetMessage(&message,NULL,0,0)
{
 if(!TranslateAccelerator(hWnd,hAccel,&message)
 {
  TranslateMessage(&message);
  DispatchMessage(&message);
 }
}
```

Ainsi, le message reçu par votre application ne poursuit son chemin dans la boucle que s'il n'existe pas d'accélérateur correspondant. Dans le cas contraire, le message est intercepté par Windows qui se charge de tous les détails de la traduction.

Il est possible, lors de la réception d'un message WM_COMMAND ou WM_SYSCOMMAND, de savoir s'il provient du choix d'une commande dans le menu ou de la traduction d'un accélérateur, en examinant les 16 bits de poids fort de lParam du message. Si cette valeur est 0, alors le message provient du menu; si elle est 1, il s'agit de la traduction d'un accélérateur.

Précisons, pour terminer, que si l'application active est sous forme d'icône, seuls les accélérateurs correspondant à un message WM_SYSCOMMAND seront traduits et envoyés à l'application.

La cuisine chinoise, ou menus et idéogrammes

Tout cela est bien joli, me direz vous, mais on peut écrire des menus semblables en mode texte et à bien moindre frais, alors qu'est ce que Windows apporte de plus ? Windows ne fait aucune espèce de différence entre du texte et des graphiques point par point (bitmaps); les deux ne sont pour lui que des groupes de pixels allumés ou éteints. Cette approche permet de définir un menu avec quasiment n'importe quoi et, en particulier, des éléments graphiques. Que diriez-vous d'un menu dans lequel la commande « Effacer » serait remplacée par un dessin représentant une poubelle? Il se trouve que Windows permet ce genre de fantaisies avec autant de

facilité que la création d'un menu conventionnel. Commençons par charger en mémoire une image point par point (nommée « Trash ») à partir du fichier ressources (tout ce code se déroule dans WinMain):

```
HBITMAP hTrash;
hTrash=LoadBitMap(hInstance,"Trash");
```

Maintenant, il ne reste plus qu'à ajouter cette image au menu de notre application. La fonction ModifyMenu() change, dans le menu hMenu, l'élément identifié par IDM_DEL pour le remplacer par notre image:

```
ModifyMenu(hMenu,IDM_DEL,MF_APPEND|MF_BITMAP,IDM_DEL,(LPSTR)MAKELONG(hTrash,0));
```

C'est tout et ça marche (la preuve: examinez le programme du mois). Bien sûr, le code est simplifié à l'extrême et ne tient pas compte des différences de résolution d'un écran à l'autre. Pour bien faire, il faudrait au préalable étirer (ou compresser) l'image pour l'adapter à la résolution de l'écran, mais ce thème serait hors sujet aujourd'hui. Nous l'aborderons au moment d'étudier les fonctions graphiques de Windows.

La cuisine nouvelle

Entre autres améliorations, Windows 3.0 permet de faire quelques nouveaux trucs avec les menus. Le fait le plus intéressant est la possibilité d'afficher un menu « pop-up » n'importe où sur l'écran. Examinons comment afficher un tel menu (« menu flottant », selon la dénomination du Kit de Développement) lorsque l'utilisateur presse le bouton droit de la souris.

La première chose à faire est d'intercepter le message WM_RBUTTONDOWN, qui est envoyé lorsque l'utilisateur presse le bouton droit. Pour cela, rien de plus simple: il suffit de modifier l'instruction switch() dans la fonction de gestion de la fenêtre dans laquelle le menu flottant doit s'afficher:

```
switch(message)
{
  ...
  case WM_RBUTTONDOWN:
     break;
  ...
}
```

Lorsque le message WM_RBUTTONDOWN est pris en charge par la fonction de gestion de la fenêtre, il faut, avant d'afficher le menu, disposer de deux éléments. D'une part, il faut connaître les coordonnées du point où se trouvait le curseur de la souris lorsque l'utilisateur a pressé le bouton. Ces coordonnées se trouvent dans le paramètre lParam du message:

```
POINT pPoint;
pPoint=MAKEPOINT(lParam);
```

Ces coordonnées, définies par rapport à l'espace client de la fenêtre, doivent être au préalable converties en coordonnées relatives à l'écran tout entier:

```
ClientToScreen(hWnd,(LPPOINT)&pPoint);
```

D'autre part, il faut disposer d'un vecteur (hMenu) sur le menu à afficher. Il peut s'agir soit d'un menu défini dans le fichier ressource, soit d'un menu dynamique construit par l'application. Il est même possible d'utiliser un sous-menu du menu principal, à condition d'obtenir un vecteur sur ce sous-menu à l'aide de la fonction GetSubMenu(). C'est la méthode utilisée dans le programme de ce mois. Il ne reste plus qu'à afficher le menu et à laisser Windows gérer les interactions avec l'utilisateur:

```
TrackPopupMenu(hMenu,NULL,pPoint.x,pPoint.y,NULL,hWnd,NULL);
```

Si le menu a été crée dynamiquement, il ne faut pas oublier de le détruire lorsque l'application se termine, par un appel à DestroyMenu().

Bruits de casseroles...

Cela fait plusieurs mois déjà que Windows 3.0 est en vente et que j'utilise le nouveau Kit de Développement, alors voici mes premières impressions. A première vue, c'est pas le pied: les trois gros (mais pratiques...) classeurs ont été remplacés par quatre volumes (plus un petit fascicule d'une quarantaine de pages qui couvre l'installation) reliés sous couverture cartonnée. Je passe sous silence la séparation du Manuel de Référence en deux volumes. Après tout, que l'information recherchée se trouve systématiquement dans l'autre volume que celui dans lequel on la cherche n'est qu'un inconvénient mineur en comparaison de la tare qui affecte ces nouveaux manuels : ils refusent obstinément de rester ouverts à plat ! Le seul remède a consisté à guillotiner le dos de la reliure, à perforer trois trous et à insérer le tout dans les anciens classeurs. Il paraît que ce type de reliure rend la photocopie plus difficile et donc serait susceptible de limiter la diffusion de copies pirates (sans blague...). Monsieur Gates, restons sérieux et revenons aux bons vieux classeurs à trous.

Rassurez-vous, il n'y a pas que des inconvénients. Le SDK 3.0 permet, comme vous vous en doutez, de développer des applications pour Windows 3.0. En dehors d'une esthétique un peu plus raffinée, Windows 3.0 est un paradis pour les programmeurs. Jugez-en plutôt par vous-mêmes. Système d'aide standardisé, utilisable à partir de n'importe quelle application (y compris la vôtre). Mode protégé sur les 80286, 80386 et 80486, avec accès à autant de méga-octets de mémoire étendue qu'il y en a dans la machine (si, si, vous avez bien lu). 16 couleurs sur les EGA/VGA (au lieu de 8) et utilisation de plusieurs couleurs dans les icônes. Nouveaux menus et dialogues, avec la possibilité d'y utiliser n'importe quel jeu de caractères (dont le jeu standard proportionnel). Les librairies dynamiques (DLLs) ont bénéficié de plusieurs améliorations.

Windows 3.0 incorpore maintenant le support des réseaux locaux. A part ça, le SDK a également subi un lifting complet. Tous les outils de développement ont été revus et améliorés. La chasse aux morpions est facilitée par un débogueur digne de ce nom: CodeView, version Windows, accompagne (enfin...) le traditionnel SYMDEB, qui commence à accuser un coup de vieux. Le programme d'installation du Kit fait même preuve d'un semblant d'intelligence et les librairies Windows ont été réécrites: les fonctions C font l'objet de librairies séparées (une par modèle de mémoire - seules, les fonctions utilisables sous Windows en font partie); la librairie

WINDOWS

Windows à proprement parler est unique et indépendante du modèle utilisé. Et, puisqu'on parle de réécriture, les manuels ont été aussi passés à la casserole - heureusement, dans le bon sens (reliure mise à part...).

Le Kit a aussi perdu pas mal de graisse au cours de ce lifting: il n'y a plus d'interface vers le MS Pascal, les versions du compilateur utilisables sont, à l'exclusion de toute autre, 5.1 et 6.0 (et Quick C 2.0) et bon nombre d'utilitaires tels que MAKE, LIB ou LINK4 ont sauté - il faut utiliser ceux fournis avec le compilateur.

A votre tour

Le programme du mois est une démonstration de quelques unes des possibilités de Windows 3.0 en ce qui concerne la création de menus. Avant de le compiler, pensez à créer une petite image au format bitmap, que vous nommerez TRASH.BMP. En attendant prochain article, où nous parlerons des boîtes de dialogue, à votre tour de suer devant la feuille blanche: décrivez-moi ce que vous faites, ce qui vous intéresse, les sujets que vous aimeriez que j'aborde (et ceux dont vous ne voulez pas entendre parler) et, en général, ce que vous attendez de Debug Magazine. Adressez votre courrier au journal en précisant sur l'enveloppe qu'il m'est destiné.

AVIS AUX ABONNES

Pour assurer la réception ininterrompue de **DEBUG MAGAZINE**, faites-nous connaître d'avance vos changements d'adresse. Ainsi, vous vous exposerez pas à une interruption accidentelle du service.

NOTE

Par manque de place, nous n'avons pu publier dans ce numéro, la rubrique **IDEE**. Nous espérons pouvoir la publier dans le numéro suivant.

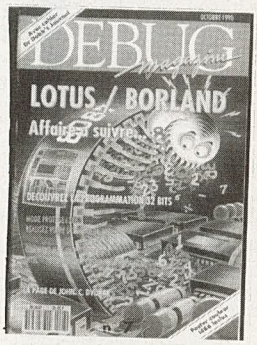
n° 2

n° 3

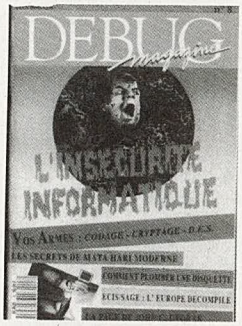
n° 4

n° 5

n° 6

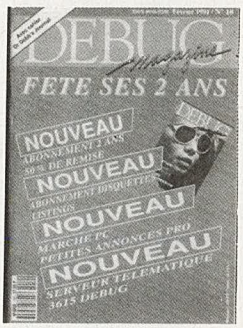


n° 7



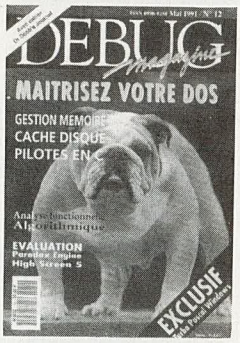
n° 8

n°9



n°10

n°11



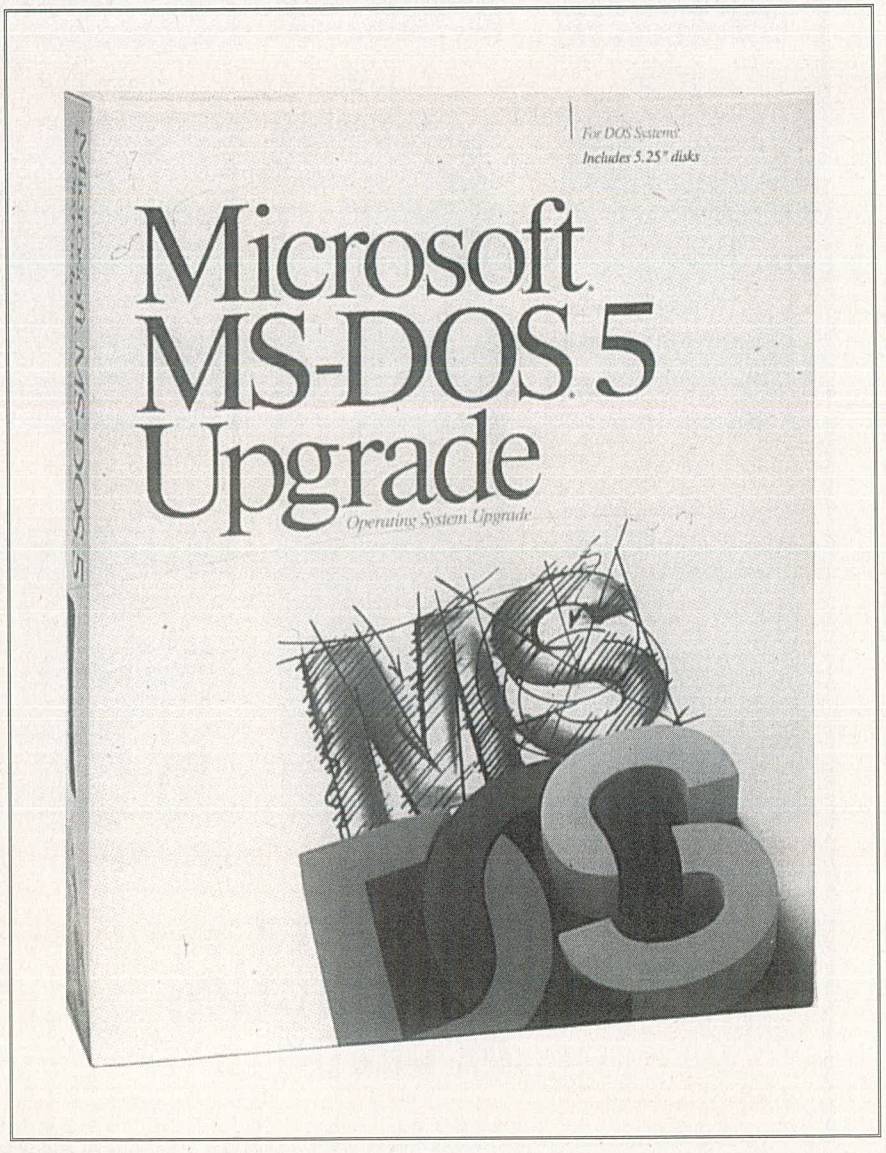
n°12

Hors-Série n° 1

COMPLÉTEZ VOTRE COLLECTION DE DEBUG

LE DOS NOUVEAU EST ARRIVE !

par *ART*



Pour la première fois dans son histoire, Microsoft va vendre le DOS en détail. Plus précisément, c'est la « mise à jour » MS-DOS version 5.0 en français qui sera disponible à partir du 15 juillet chez Microsoft et chez ses revendeurs. Longtemps attendu, MS-DOS 5 est arrivé.

[SERVICE LECTEURS : INTERESSANT : 31 / MOYEN : 32 / MAUVAIS : 33]

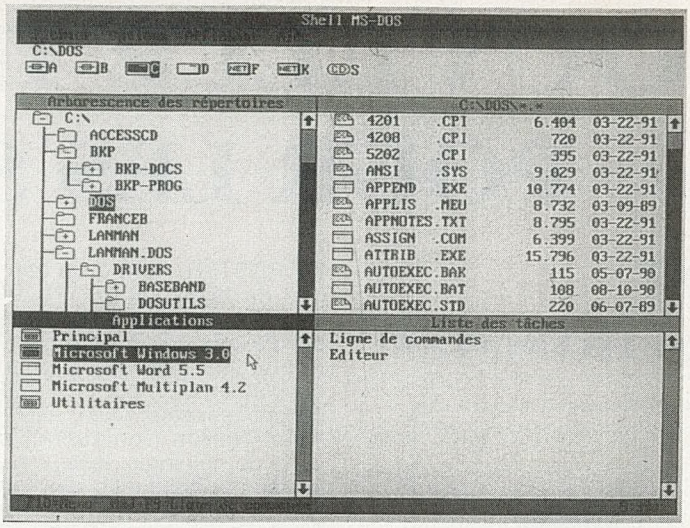
MS - DOS 5

Le temps d'attente a été fructueusement employé à tester la version bêta parmi 7000 sites de test, dont 400 en France. Selon Bill Gates, qui a présenté la version américaine le 11 juin dernier à Boston, « MS-DOS 5 est sans doute le logiciel ayant subi les tests les plus approfondis de toute l'histoire de l'informatique, ce qui garantit sa fiabilité et stabilité ».

Actuellement, l'usine en Irlande tourne à plein régime en fabriquant des versions localisées : française, allemande, arabe, russe, hébreu, etc... Actuellement, il y a 16 versions nationales prévues (manuel et langage des messages). Chaque version peut être en plus configurée conformément aux 24 autres normes ou claviers (tchèque, slovaque, hongrois, polonais, suisse romain ou germanique etc.). Selon Microsoft, quelque 60 millions d'utilisateurs du DOS dans le monde sont susceptibles d'être intéressés par l'installation de cette nouvelle version à la place du DOS actuel.

MS-DOS 5.0 se distingue par 4 points forts : meilleure gestion de la mémoire, interface d'utilisateur plus conviviale, jeu d'outils complémentaires et la capacité de remplacer toutes les versions précédentes.

D'abord la mémoire : depuis longtemps il n'est plus secret que des programmes nouveaux (comme HIMEM, EMM386 ou LOADHIGH) permettent d'exploiter toute la mémoire centrale disponible. Le noyau du DOS lui-même peut se reloger dans la mémoire haute (au delà d'un Mo), ce qui peut libérer 45 Ko sur les systèmes basés sur 286 et davantage sur 386/486. L'exécution

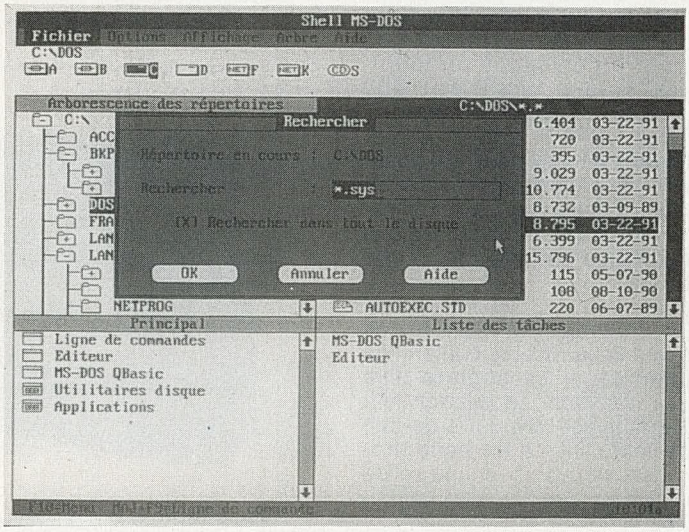


Vue graphique du système DOS

en mémoire haute n'est pas possible sur les processeurs 8088 et 8086, incapables d'adresses plus d'un mega-octet de la mémoire. MS-DOS 5 sait aussi gérer les partitions de plus de 32 Mo (jusqu'à 2 Go), ce qui libère de la mémoire occupée éventuellement par des pilotes spéciaux ou la commande SHARE. Une nouveauté: le nouveau format de disquettes 2.88 Mo.

Certains utilitaires, qu'on devait acheter à part, sont livrés avec MS-DOS 5. Sous licence Central Point Software, Microsoft à inclus les outils de récupération UNDELETE et UNFORMAT et le programme de formatage rapide de disquettes de la version 6 de PC TOOLS. Un nouveau SHELL, complètement réécrit aux normes SAA, permet de manipuler les fichiers à l'aide des icônes à la Windows 3 et lancer les programmes sans avoir à frapper les noms des fichiers sur le clavier. Nouveauté : un commutateur de tâches permettant de changer rapidement des programmes par sauvegarde de la mémoire sur le disque dur. Enfin, chaque exemplaire de MS DOS 5.0 est accompagné par l'interpréteur QBasic et son environnement intégré de développement (éditeur écran pleine page, débogueur et documentation en ligne).

Alors, acheter ou mettre à jour? Microsoft et ses revendeurs ne vendent qu'une « mise à jour » (inutilisable directement car non « bootable »), la version générique (« auto-bootable ») sera vendue seulement par des constructeurs des machines (dont 130 ont déjà signé, selon Microsoft, l'accord de la licence) - un prix de vente en détail n'existe même pas. L'ensemble « Mise à Jour » sera vendue au prix de 790 Frs TTC. Il existe aussi un ensemble appelé MLP (Microsoft Licence Pack), contenant seulement la disquette matrice, sans manuels, permettant d'effectuer des installations dans le cadre des licences site. Le droit de copie MLP (que certains appellent « Mouvement de Lutte anti-Pirates »...) vaut 632 Frs TTC.



Fonctions évoluées

Ce qu'il faut souligner est que Microsoft n'exige aucun document ni preuve d'achat pour pouvoir acheter l'ensemble « Mise à Jour ». Curieusement, on retrouve ainsi une certaine similitude avec l'esprit shareware : « essayez et quand vous voulez utiliser, achetez ». Alors, si vous êtes content de votre MS-DOS actuel et si vous voulez avoir un MS-DOS le plus récent, à vous de jouer. Et « Qu'aucun PC ne reste sans! », selon le slogan publicitaire américain.

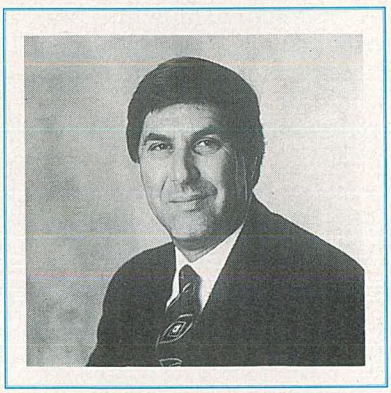
NOTE : disposant maintenant de la version définitive, DEBUG MAGAZINE va pouvoir publier prochainement les tests comparatifs.

NOUVELLE ENERGIE POUR LES STARS

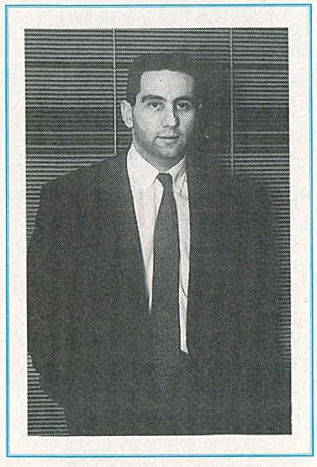
par *ART*

LES HOMMES

Yann de Pontbriand



Ron Posner



Laurent Massa

WORDSTAR, dont on a peu entendu ces derniers temps, revient avec l'énergie nouvelle. Une nouvelle équipe, dirigée en France par Yann de Pontbriand (issu de la galaxie Sun) épaulée en USA par Ron Posner (précédemment chez Ansa, Ashton Tate et Peter Norton Computing) s'installe dans de nouveaux locaux et présente la gamme habituelle des logiciels de traitement de texte enrichie de quelques nouveautés : WordStar 2000 version 3.5, WordStar 6.0 et Wordstar Legacy (version Windows, acheté chez NBI). La ligne de produits pour WINDOWS sera étendue par Correct Grammar pour Windows (de chez Lifetree Software), un vérificateur orthographique et stylistique et Wordstar pour Windows, actuellement en développement.

Dans WordStar 6.0 les amateurs et des utilisateurs habitués retrouveront avec plaisir l'interface classique Wordstar. Ceux qui préfèrent des présentations plus « modernes », vont pouvoir utiliser les menus déroulants à la norme SAA. Le logiciel arrive accompagné d'Inset, programme de capture graphique, MailList, outil de publipostage, ProFinder, gestionnaire de fichiers sur disque, StarExchange, programme de conversion de fichiers et TelMerge - outil de télécommunication X-Modem/Y-Modem.

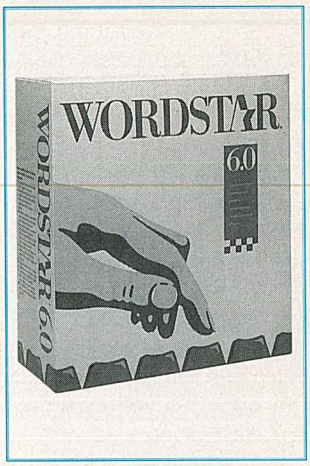
Une nouvelle collection de logiciels voit aussi le jour : Wordstar Laptop Collection : les logiciels « allégés », apportant des fonctionnalités spécifiques, utiles aux utilisateurs des portables comme le transfert de documents par câble ou un agenda-planning résident.

Si vous êtes l'utilisateur d'une version de WordStar et vous voulez bénéficier des nouvelles version, profitez des mises à jour à conditions préférentielles : 990 F. pour Wordstar 2 000 (4 800 F.), 1 250 F. pour WordStar 6.0 (4 500 F.) et 1 950 F. pour Wordstar Legacy (4 950 F.) - prix HT de vente des produits génériques marqués entre parenthèses. En pratique, tout utilisateur d'un WordStar peut accéder à toute version plus performante, à une fraction du prix.

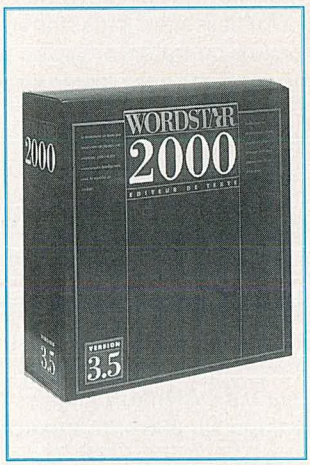
Selon Ron Posner, que nous avons rencontré lors de la récente conférence SPA à Cannes, « La société a le potentiel de réaffirmer sa position leader dans le domaine des outils de traitement de texte. Il y a de bons produits, des gens nouveaux et une solide base de plusieurs millions d' utilisateurs fidèles ».

WordStar France Immeuble Nungesser
13 av Morane Saulnier -78140 Velizy
Tél : (1) 34659081 fax (1) 30 70 63 67

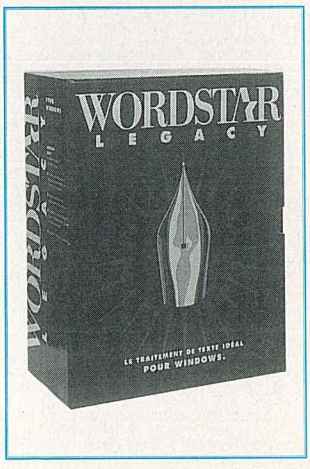
LES PRODUITS



Wordstar 6.0



Wordstar 2000 3.5



Wordstar Legacy

[SERVICE LECTEURS :INTERESSANT :34 / MOYEN : 35 / MAUVAIS : 36]

ABONNEMENT 1 AN DISQUETTES LISTING

❒ Je m'abonne pour **12** disquettes listing : **495 F** au lieu de **564 F**
Je règle par : ❒ chèque à l'ordre de **Debug Magazine**
❒ carte bancaire

10 %

Nom de la carte : .....
N° ..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../ Date d'expiration /..../..../..../..../
Nom :..... Prénom : .....
Société :.....
Rue :.....
Lieu :.....
Code postal :..... Ville : .....
Pays :..... Tél : .....
Date :..... Signature : .....

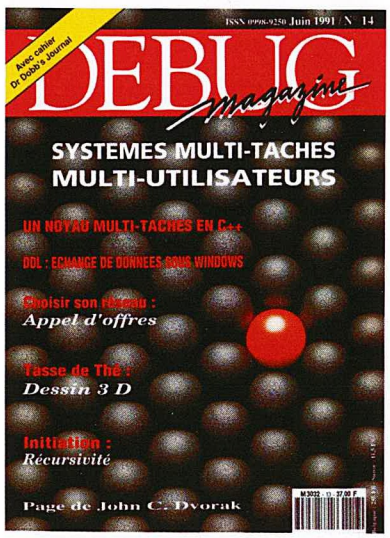
A renvoyer à **DEBUG MAGAZINE** - 14, rue de l'Atlas - 75019 Paris

COMMANDE DISQUETTE LISTING N° 14

❒ Je désire recevoir la disquette listing **DEBUG 14** au prix de **47 F** pièce

Je règle par : ❒ chèque à l'ordre de **Debug Magazine**
❒ contre remboursement (ajouter **40 F** de frais)
❒ carte bancaire

Nom de la carte : .....
N° ..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../ Date d'expiration /..../..../..../..../
Nom :..... Prénom : .....
Société :.....
Rue :.....
Lieu :.....
Code postal :..... Ville : .....
Pays :..... Tél : .....
Date :..... Signature : .....



Anciennes disquettes à **47 F:** ❒ n° 1 ❒ n° 2 ❒ n° 3 ❒ n° 4 ❒ n° 5 ❒ n° 6 ❒ n° 7 ❒ n° 8 ❒ n° 9 ❒ n° 10 ❒ n° 11 ❒ n° 12 ❒ Hors - Série n° 1
Disquettes utiliraires : ❒ PKPAK ❒ LHARC ❒ LZE, LZESHELL

A renvoyer à **DEBUG MAGAZINE** - 14, rue de l'Atlas - 75019 Paris

SERVICE LECTEURS

DEBUG Magazine n° 14

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Nom : ..... Prénom : .....
Société : .....
Rue : .....
Lieu : .....
Code postal : ..... Ville : .....
Pays : ..... Téléphone : .....

A REMPLIR OBLIGATOIREMENT SI VOUS AVEZ DEMANDÉ UNE DOC.

Le Service Lecteurs de DEBUG Magazine transmettra aux intéressés les demandes d'envoi de documentation dont vous aurez coché les cases.

Notre service vous permet également de donner votre opinion sur les articles parus. Cochez les cases correspondant à votre jugement sur l'article concerné (les petits numéros au début de chaque article, pas les numéros des pages) ; cela nous permettra d'améliorer la qualité de la revue. Merci !

pliez ici en second et collez l'autre côté au ruban adhésif.

AFFRANCHIR
AU TARIF
LETTRE..

DEBUG MAGAZINE
14, rue de l'Atlas
75019 Paris

pliez ici en premier.

QUESTIONNAIRE – LECTEUR

Année de naissance : 19__
Sexe : ☐ M ☐ F

Votre secteur d'activité :

- ☐ Agriculture
- ☐ Enseignement
- ☐ Autres Fonction Publique, Collectivité locale
- ☐ Association
- ☐ Energie
- ☐ Métallurgie
- ☐ Chimie, verre, plastique
- ☐ Construction automobile, mécanique, navale, aéronautique
- ☐ Electronique
- ☐ Bâtiment, travaux publics
- ☐ Autres secteurs industriels
- ☐ Commerce
- ☐ Transports
- ☐ Télécommunications
- ☐ SSII
- ☐ Autres services aux entreprises
- ☐ Hôtellerie, réparation et autres services aux particuliers
- ☐ Banque, Assurance
- ☐ Autres secteurs commerciaux

Nombre de salariés dans l'établissement : ____

Votre profession :

- ☐ Propriétaire
- ☐ PDG-DG-Gérant
- ☐ Cadre
- ☐ Employé
- ☐ Ouvrier
- ☐ Retraité
- ☐ Autre

Votre matériel :

- ☐ IBM ou compatible
- ☐ Apple, MAC
- ☐ Mini
- ☐ Mainframe

L'utilisation de votre matériel :

- ☐ privée
- ☐ professionnelle
- ☐ les deux

Etes-vous satisfait de votre équipement actuel ? ☐ Oui ☐ Non
Allez-vous l'étendre ? ☐ Oui ☐ Non

Vos commentaires :

APPEL D'OFFRES

tables. CID décide d'engager bientôt une télévendeuse, dont la tâche sera de prendre des commandes au téléphone et de les enregistrer directement sur l'ordinateur. Un technicien de livraison et fabrication sera aussi engagé, pour suivre les livraisons et surveiller le confectionnement des emballages. Jean Courtebotte a besoin d'accéder aux comptes, pour ses tableaux financiers prévisionnels. André, le vendeur voudrait pouvoir exploiter directement le fichier de clients pour ses mailings ciblés.

Courtebotte Industrie

UN SYSTEME MULTIPOSTE DEVIENT LA NECESSITE

Compte tenu de liens permanents entre les tâches à exécuter et les personnes exécutantes, il n'est plus possible de tout faire sur le seul ordinateur de la secrétaire et échanger les fichiers en jonglant avec des disquettes. Le premier bilan ressort un résultat positif, un montant sera alloué à l'équipement informatique musclé. Jean passe plusieurs nuits blanches à préparer son « cahier de charges ». D'abord il commence par regrouper toutes les taches que le système doit exécuter. Au fur et à mesure de ses réflexions, il devient de plus en plus évident qu'une séparation rigide entre les tâches à exécuter en commun et les tâches « personnelles » n'est pas possible. Il devient certain que le système comptable et commercial doit être multiposte : le client peut téléphoner pour savoir où est sa commande, au même moment le comptable peut enregistrer son paiement et le magasinier imprimer le bon de livraison - trois opérateurs doivent pouvoir accéder simultanément à la même transaction commerciale ou l'écriture comptable. Mais le fichier des clients, qui fait partie du système commercial multiposte sera utilisé aussi par André pour créer les mailings personnalisés. On peut juste « prélever » les adresses dans le fichier commercial, le traitement successif ne sera effectué qu'ultérieurement, par une personne seule. Par contre, la possibilité d'échange des documents est essentielle : le projet des lettres aux clients sont élaborés entre Jean et André et révisées par la secrétaire (les fâuuutès!).

L'OUVERTURE

Un nouveau besoin se fait sentir. André, soulagé dans ses obligations quotidiennes par la nouvelle équipe commerciale, va pouvoir prospecter à l'extérieur : les boutiques de luxe, supermarchés qui désirent ouvrir un rayon de vins de luxe et, évidement, le marché américain : il veut exporter. Lors de tous ces déplacements, André doit rester en contact permanent avec l'entreprise et pouvoir accéder à son système informatique. Cela est particulièrement utile lors de déplacements outre-atlantique, en Californie : le décalage horaire de 9 heures rend pratiquement impossible le contact vocal. D'autre part, la création d'une boutique à Paris est envisagée, connecté en temps partagé sur le système de la maison mère bordelaise. En plus, tout le monde veut accéder aux bases de données et fichiers extérieurs, comme les réseaux d'information ou le plus gros fichiers d'adresse : le 11 du Minitel. Le système doit être multiposte localement et à distance !

LE TEMPS PRESSE ET LE CHAOS MONTE

Jean ne sait pas si tout cela sera possible à réaliser. Il passe beaucoup de temps à réfléchir, poser des questions, comprendre, imaginer des solutions. Et pourtant, le temps presse. Le premier télévendeur commence à prendre des commandes au téléphone (sur des bouts de papier), la secrétaire, constamment pressée et fatiguée commence à pâlir, Jean « rattrape » tout le soir et rentre à la maison de plus tard. André téléphone des USA avec des nouveaux contrats et s'inquiète des capacités de livraison. Les retards s'accumulent. Jean se documente, achète des revues, va aux expositions. Les réponses ne sont pas évidentes, l'horizon devient de plus en plus éloigné et l'image se brouille. Ces ordinateurs qu'on dit personnels, ont l'air d'être de plus complexes. Comprendre, assembler un système cohérent, même se servir de certains produits reste l'affaire des spécialistes, contrairement à de premières expériences ponctuelles. Acheter n'est pas une partie de plaisir non plus. La confusion est à son comble quand on commence d'évoquer les « multi » divers, topologies, protocoles, standards ou les temps de réponse. Les revendeurs parlent d'UNIX, réseau, OS/2, DOS. Quelle est l'avantage d'une station de travail. Et qu'est le RISC ? Pour ne pas risquer de se tromper, pressé par le temps, Jean trouve enfin une solution miracle. Il s'est rappelé son vieil ami qui « bidouille » dans l'informatique, il l'invite à un dîner dans le fameux restaurant La Plaisance à Saint Emillion et le projet d'un Cahier de Charges voit le jour (au petit matin...).

CAHIER DE CHARGES, OU PLUTOT « UN APPEL D'OFFRE AUX FOURNISSEURS »

Le système informatique de Courtebotte Industrie et Distribution SA sera multiposte, compatible, ouvert, évolutif et performant.

Il doit supporter les diverses applications « standard » mono et multipostes, parmi lesquelles le traitement de texte, base de données, tableur, PAO et dessin assisté, comptabilité et gestion commerciale intégrés et fonctions de télécommunication par texte, voix et image (modem, fax, messagerie vocale et Minitel). Il doit être compatible avec les équipements compatibles IBM PC/AT et Mac.

La configuration initiale sera composée d' équipements actuels qu'on va relier ensemble et des équipements nouveaux, achetés pour répondre aux tâches supplémentaires.

DEUX CONFIGURATIONS ET DES OPTIONS

La proposition doit chiffrer deux configurations fondamentales et certaines options. Une première configuration doit disposer de 5 postes de travail, dont 3 existent déjà (un AT 386, un AT 286 et un Mac, mais il faut les chiffrer). Deux postes seront achetés pour le service commercial et comptabilité, ces deux postes peuvent être dépourvus des disques locaux. La deuxième configuration comportera 20 postes de travail (dont 10 peuvent être éventuellement dépourvus des disques locaux)

et 3 doivent être des Macs. Tous les postes seront équipés des écrans couleurs VGA, mémoire centrale de 2 Mo et vitesse d'au moins de 20 MHz.

Le serveur de fichiers doit disposer d'au moins de 400 Mo de capacité de stockage (éventuellement divisé en deux volumes de 200 Mo) qu'on portera ultérieurement à 1 Go. Le serveur doit pouvoir stocker et manipuler des fichiers texte et binaires, avec les fonctions de partage et verrouillage d'un ensemble de fichiers (arborescence des répertoires), d'un fichier et d'un enregistrement. Au moins à partir d'un poste, on doit pouvoir lire et écrire tous formats des disquettes PC et Apple.

Le système offrira à tous les utilisateurs une imprimante rapide (au moins 300 cps, 132 car) multifonction pour le papier en continu et une imprimante laser (format A4), minimum 8 pages/minute, 300x300 dpi, avec option POSTSCRIPT, pour les impressions de qualité et graphiques. Chaque station doit pouvoir imprimer sur des imprimantes communes, on doit éventuellement pouvoir connecter une imprimante en local. Toutes les stations doivent pouvoir disposer d'une souris et d'une connexion série libre.

COMPATIBILITE ET EVOLUTION

Le système doit être parfaitement compatible avec les produits standards, disponibles en commerce, notamment MSDOS, NetBios et Système 7. Les postes à acheter seront équipés des processeurs modernes, au moins des 386SX, doivent êtres compatibles WINDOWS pour les compatibles PC et des processeurs 68030, être compatibles Système 7, pour Mac. Les stations de travail doivent avoir la capacité d'extension multitâche. Bien que le système ne sera pas étendu dans l'avenir proche au delà de 20 connexions simultanées, on doit prévoir une extension éventuelle vers un système à 50 postes, avec 250 postes comme le plafond limite. On précisera le coût de ces extensions, par module (ensemble) standard.

OUVERTURE

CID SA est une entreprise commerciale, la communication efficace est sa raison d'être. Chaque fonctionnalité sera chiffrée à part, comme une option. Le personnel de CID doit pouvoir communiquer en interne (échanger des textes, fichiers et messages) et communiquer avec les ressources extérieures telles que les services Minitel et réseaux spécialisés accessibles par TRANSPAC, comme des messageries électroniques au standard X.400. La vitesse de communication doit être de 1200 à 9600 bauds (V21 à V32), avec protocole V42 (NMP 1 à 5). Une passerelle permettant de se connecter en 3270 sur un système IBM 3090 équipé d'un 3745 frontal et sur un VAX 8700 en VT100/220, situés à la Chambre de Commerce des Industries Vinicoles à Saint Emillion, sera chiffré.

Le système doit permettre de recevoir et envoyer les télécopies (avec fonction de publipostage automatique) à partir de chaque poste de travail. Éventuellement, la fonction de messagerie vocale, connectée sur le standard téléphonique, sera proposée. Le système doit permettre de se connecter de l'extérieur, d'une part par l'autre système indépendant et similaire situé à l'antenne parisienne, d'autre part par un portable itinérant (dans le monde entier), équipé d'un modem.

Les clients de CID SA possèdent tous les Minitels. Ils doivent pouvoir disposer d'un serveur Minitel (3615 VIN, 16 lignes simultanées), permettant notamment de consulter le catalogue et placer des commandes. Le responsable de communication chez CID doit pouvoir intervenir sur ce serveur à partir de son poste de travail, pour réaliser les pages et arborescence ainsi que récupérer les fichiers crées par les utilisateurs de service. La proposition chiffrera seulement le coût de l'équipement (matériel et logiciel) nécessaires pour installer un tel serveur, sans sa création. Chaque utilisateur doit pouvoir accéder de son poste à chaque imprimante, disque ou processeur connectés au système (lancer un programme), dans la limite de ses autorisations d'accès.

PERFORMANCES ET RÉSULTATS

Les performances d'une station connectée sur le système doivent être comparables aux performances atteintes en local, comparés dans le contexte identique. La connexion au système ne doit pas pénaliser une station de travail ni du point de vue de vitesse ni encombrement de mémoire.

LA SECURITE

La sécurité sera un point primordial de l'offre. Les serveurs dédiés vont être préférés aux serveurs partagés, à cause de possibilité d'un « plantage » par un programme lancé en local ou débranchement accidentel, catastrophique pour l'ensemble de tous les utilisateurs. Les dispositifs de sécurité doivent garantir un niveau raisonnable de confidentialité. Les serveurs, postes de travail et ressources ne vont pas être accessibles que par les opérateurs autorisés. Sur le serveur de fichiers, il sera désigné des espaces (répertoires) communs et des espaces privatifs. Seul le superviseur sera autorisé d'accéder à tous les espaces.

Un outil de détection et prévention de virus sera prévu. Un système de sauvegarde des stations locales sur le serveur de fichiers et la sauvegarde automatique du serveur sur des médias amovibles, qu'on peut emporter en lieu sur, sera prévu. Un onduleur garantira automatiquement le temps de sauvegarde du serveur, en cas de panne du secteur.

Pour faciliter les manipulations par tous les opérateurs à partir de tous les points du système et garantir la redondance de ressources, les éléments du système doivent être interchangeables. Le système doit être accessible pour tous à partir de chaque station de travail, tout en garantissant le contrôle d'accès et privilèges. Une interface standardisée facilitera les manipulations de l'ensemble par du personnel non spécialisé. Le système doit pouvoir globalement fonctionner même en cas de panne d'un sous-ensemble. Dans le cas de panne majeure du serveur, le temps maximal de redémarrage (y compris la reconstitution du contenu) ne doit pas dépasser 48 heures.

LES SERVICES ET GARANTIES

La proposition doit chiffrer le coût d'installation sur site, « clef en main », câblage inclus. Une formation de l'administrateur du système et utilisateurs est nécessaire.

Le système doit être totalement garanti pendant au moins 3 mois après l'installation, le temps de rodage. Au delà, un contrat de maintenance sur site sera proposé, garantissant le dépannage en 48 heures ouvrables. Le superviseur du système doit pouvoir disposer des outils permettant de suivre le fonctionnement de l'ensemble, l'activité et l'usage de ressources, détecter des noeuds d'étranglement et pannes éventuelles. Les soumissionnaires sont invités à fournir des références d'installations opérationnelles. Une visite sur site, avec une démonstration éventuelle, est à prévoir.

RUMEURS

...Suite de la p. 12

Évidemment, aucun droit d'utilisateur, protection contre les défauts, malfonctionnements ou l'abandon n'a été instaurée, cela n'a même pas fait l'objet de débats. L'utilisateur a droit de « faire une copie de sauvegarde dans la mesure où elle est nécessaire pour l'utilisation ». Comment va t'on juger si une sauvegarde est nécessaire ou non? La directive ne réponde pas. Et comment faire si le logiciel est plombé par une disquette ou dongle, surtout lorsque « la mise en circulation et détention à des fins commerciales de tout moyen ayant pour seul but de faciliter la suppression non autorisée ou la neutralisation de tout dispositif technique éventuellement mis en place pour protéger un programme » devient illégale. Adieu CopyWrite ou autre CopyIIPC.

La CEE va devenir une enclave de plombages, au détriment du bon sens et des intérêts de tous. Comment est-on arrivé à un tel non-sens ? La procédure bruxelloise a fait les siennes : pour adopter une proposition la majorité de 50% est requise. Le vote avait lieu le soir, en absence de la majorité des parlementaires. Bien que la majorité (75%) de présents a été atteinte, à cause des absents la proportion de votants n'a pas suffi pour dépasser les 50% de la totalité des votes. C'est ainsi que la CEE fait ses lois, c'est ainsi qu'elle s'est tissée elle même la corde par laquelle elle risque d'être un jour étranglée. Triste résultat!

Terminons sur une note plus positive : selon une enquête DATAPRO durant COMDEX Spring 1991, 72% de 500 interrogés ont déclaré que leurs achats des ordinateurs vont augmenter pendant les 12 mois à venir. 78% vont dépenser davantage pour le logiciel et 65% pour les Réseaux Locaux. Les perspectives sont meilleures pour les vendeurs des Réseaux : 39% des répondants ont déclaré que la technologie des Réseaux va avoir l'impact le plus significatif sur leur activité. 28% ont cité Windows et seulement 8% UNIX.

Alors, où est cette crise?

Faites-vous connaitre.
De temps en temps, nous louons nos fichiers à divers annonceurs, désirant communiquer directement avec nos lecteurs. Si vous ne désirez pas recevoir directement ce type de messages publicitaires, veuillez nous le faire savoir. Nous allons désactiver votre adresse dans nos fichiers.

Un noyau multitâche en C++

...Suite de la p. 15

Pour approfondir la question de communication entre tâches à la Modula-2, je vous recommande le chapitre 16 du livre « Modula-2 : A Software Development Approach ».

Il me reste quelques suggestions quant à l'amélioration de ce noyau. Attribuer une priorité à une tâche peut constituer une propriété commode. Il suffit d'adjoindre une *ready_q* à l'objet *task_control* pour chaque niveau de priorité, ou de se servir d'un tableau de files *ready_q*. Il ne faut pas avoir recours à une très large collection de niveaux de priorité : cinq paraissent suffire. La procédure *task_switch* examine les files *ready_q*, la priorité la plus élevée en premier, pour trouver la prochaine tâche à activer. Au moment d'ajouter une tâche à une *ready_q*, il suffit alors d'évaluer sa priorité pour la mettre dans la bonne file.

Intégrer le moyen pour une tâche de se mettre en sommeil, pendant un nombre donné de tops d'horloge, peut aussi être fort utile. C'est un peu plus délicat, et vous pouvez aborder la question de différentes manières. Je pense, toutefois, que vous constaterez qu'il est très simple d'ajouter ces propriétés au noyau, ou de l'adapter à votre guise.

Biographie

Ford, Gary A. et **Weiner, Richard S.**
« Modula-2: A Software Developement Approach »
John Wiley & Sons, New York 1985

Weiner, RichardS. et **Pinson, Lewis J.**
« An Introduction to Object-Oriented Programming and C++ »
Addison-Wesley, Reading, Mass. 1988

Tom Green est ingénieur
à Central Data Corp.
à Champaign, I11.
Il peut être contacté au :
19. 1 (217) 359 8010

Un noyau Multi-tâches en C++ :
copyright **Dr Dobb's Journal**
M&T Publishing,
Inc, RedwoodCity,
CA USA 1990.
All right reserved.

Un noyau multitâche fondé sur le principe de « préséance ».

...Suite de la p. 23

En général, j'use de la sortie vidéo directe dans les applications multitâches car entrer dans le DOS est trop aléatoire (il emmêle le système d'interruptions). Parce que les deux tâches sont de même priorité, elles vont s'exécuter à tour de rôle - cinq secondes durant, des 1 et des 2 s'afficheront alternativement.

Notes

1. NdT. « bug »: puce de mer. Petit insecte sauteur et translucide qu'on rencontre sur les plages. J'ai une nette préfèrence à traduire « debug » par épucer. Voir dans « Mon Faust », Paul Valéry, NRF, acte 2 : «...et tout ce petit monde qui fait de l'opposition ». Un abonnement gratuit d'un an est offert par la revue à celui qui trouve la scène d'où est extraite cette citation. La présente note n'est pas qu'un caprice culturel : elle tient compte d'observations sagaces relevées dans le courrier des lecteurs.

2. cf. Diderot « Dieu est le grand ordonnateur du monde ». Celui qui commande, organise, arrange. L'anglais « scheduler » désigne celui qui établit un horaire, un calendrier.

Un noyau multi-tâches fondé sur le principe de « préséance » : copyright **Dr Dobb's Journal**
M&T Publishing,
Inc, Redwood City,
CA USA 1990
All right reserved.

DEBUG MAGAZINE
s'intéresse à vous.
Nous aimerions recevoir des informations sur les nouvelles technologies, les nouveaux produits ou les développements de votre entreprise. Envoyez-nous vos communiqués de presse, par courrier ou télécopie.
Et si vous voulez donner un coup de projecteur sur une info (ou une vacherie fondée à dire) ne vous gênez pas.

```
Assign(Cliche,Nom_Fichier);
Reset(Cliche);
P_Cliche:=Ptr($A000,$0); {début de la page vidéo}
For I:=$0 to $3 do {pour chacun des 4 plans}
  Begin
    PortW[$03C4]:=2+(($01 shl I)*256); {activer du plan I}
    Read(Cliche,P_Cliche^) {et lire l'image}
  End;
Close(Cliche);
```

Ces deux routines supposent que les modes de lecture et d'écriture de la carte vidéo sont les modes par défaut du BIOS (à savoir, 0 dans les deux cas). Cependant, comme les pilotes BGI de Borland ne modifient pas ces modes de manière permanente, vous n'avez pas à vous en soucier. Si vous désirez en apprendre plus sur la programmation des cartes EGA et VGA, je vous conseille deux ouvrages (parmi beaucoup d'autres): « EGA/VGA: A Programmer's Reference Guide », par Bradley D. Kliewer, édité chez McGraw-Hill (en anglais); « Guide Microsoft de la programmation des cartes graphiques », par Richard Wilton, édité chez Microsoft Press (en anglais) ou chez Sybex (en français). Cette liste est loin d'être exhaustive ou de citer ce qui se fait de mieux, mais je n'ai que ces deux titres sous la main.
Serge Grun

DÉCOMPRESSONS !

☛ ***« Lecteur occasionnel, ayant lu votre n° 12, j'ai été surpris de voir que les disquettes sont toujours proposées à la vente mais sans aucun avertissement indiquant leur utilisation à travers de compresseur de Backup Pro. D'ailleurs, votre revue ne fait plus référence à ce programme de sauvegarde et ne le distribue plus à ses abonnés. Aurais-je encore une fois manqué d'information essentielle ou est-ce à dire que ces disquettes sont maintenant en clair? Dans le cas contraire, je me demande comment les acheteurs sont censés se débrouiller pour récupérer les fichiers ».***
Jean Marc Brassard, Saint Laurent du Var

Seules les disquettes du N° 9, Hors Série N° 1 et Backup PRO (qui n'est plus disponible, nous avons distribué gratuitement à nos abonnés la totalité de 1 500 que nous avons acheté) sont compressées - le volume de fichiers dépasse 360Ko. L'outil de compression utilisé est LHarc de Yoshi et non Backup PRO. LHarc est un freeware, vous pouvez le trouver dans toutes les bibliothèques du domaine public, il est gratuit. Vous pouvez aussi acquérir la copie chez nous. En plus, depuis quelque jours, nous l'avons inclus sur ces disquettes, sous forme auto-extractable. En clair, maintenant les fichiers s'extraient tout seuls, parmi eux LHarc et son manuel!

1° - JE NE PLOMBERAI POINT

☛ ***« Nous partageons entièrement les propos cités dans la documentation de Backup PRO. Nos logiciels ne sont pas protégés contre la copie et l'achat d'un de nos logiciels vous autorise à vous en servir sur tous les micros de votre site. D'une part la Loi vous autorise à faire une copie de sauvegarde, les logiciels « plombés » violent cette loi et si d'autre part leur plombage n'est pas mentionné au moment de l'achat, vous pouvez exiger de votre vendeur le remboursement de logiciel. Il s'agit d'une caractéristique essentielle du logiciel pouvant influencer l'achat dans un sens ou dans l'autre, cacher cette caractéristique majeure s'appelle du vol. Nous saluons ici le transportable GOLF de GOUPIL qui allie beauté, puissance et efficacité ».***
Thierry Cumps, Saint Sever

RECHERCHE DE SPONSOR

☛ ***« J'ai 17 ans, je suis très intéressé par l'informatique et mes parents m'aident, comme ils peuvent et selon leur moyens, dans cette tâche. Mes parents étant des simples ouvriers, je ne peux leur demander de faire plus pour moi, qu'ils ne le font. Ce Noël, ils se sont privés pour m'acheter un ordinateur compatible tout en sachant que leur budget ne leur permettait pas. Ma lettre m'amène donc à vous demander s'il vous serait possible de m'adresser à titre gracieux, certains de vos matériels (livres, logiciels etc.) dont vous n'avez plus d'utilité ».***
S.S., Plaisir (contact à la Redaction)

PLOMBAGE VOLÉ, SERVEUR MUET

3614 DEBUG ne répond pas!
Et il ne va pas répondre
encore pendant
quelque temps.

La raison ?
Il y en a deux :
d'abord,
nous avons attendu
longtemps l'attribution
du code abrégé
PTT -
l'histoire
des formulaires
périmés fournis
par un service
est inacceptable
pour un autre,
tampons manquants, etc.

Une fois que tout a été
réalisé, bouclé et complété,
la veille du démarrage
un voleur mysterieux
nous a subtilisé
les deux dongles
« plombant »
les logiciels
du serveur.
En effet,
nous avons prévu
d'utiliser
le logiciel HOSTEL
et Kristel II de Goto
informatique.

Les deux sont « plombés »
par les dongles.
Le vol de dongles bloque
les logiciel et nous empêche
de démarrer le service.

Nous devons donc passer
quelque 48.000 Frs TTC
(coût de logiciels et de
la carte X25) en pertes
et profits (l'assurance ne
rembourse pas car il n'y avait
pas d'infraction) et tout est à
recommencer de nouveau.
Si vous « tombez »
sur HOSTEL
N° série 160225
ou KRISTEL II
N° série 6150364,
alertez-nous.

Cela nous intéresse
bigrement !

LE PROCHAIN NUMERO

Debug Magazine n° 15 (juillet / août 1991)

Parution vers le 15 juillet 1991

Cahier Dr Dobb's Journal :
- Créer son propre extendeur du DOS par Al Williams
- Alternative à SOUNDEX par Jim Howell
- Une interface graphique standard pour WINDOWS, PM et autres GUI.

Et Aussi :
- Affichage des répertoires par Gerard Foucault
- Compléteur des commandes DOS par Olivier El Guedj
- Gestion de la souris en Pascal par Olivier Pecheux
- Initiation : Comment construire un programme par J Ch Riat
- Initiation : Comment implémenter le polymorphisme dynamique en ASM par Paul Francheschi

Et comme d'habitude :

- Podium (!!!)
- Idée (???)
- Page de John C Dvorak

SONDAGE PERMANENT

Nos lecteurs ont attribué les scores suivants aux articles parus dans **DEBUG MAGAZINE n° 12:**

1° Des arbres programmatiques par J Ch Riat
2° Gestionnaire de Périphériques par A Klein (DDJ)
3° Rumeurs par *ART*

Nous félicitons les auteurs!

Nous avons les premiers résultats obtenus par les articles parus dans **DEBUG MAGAZINE Hors Série n° 1:**

1° PUT par M Canal et Presse Disquettes par F. del Greco
2° Turbo Ecriture par G. Cubizolle
3° Visualiser les caractères par J Oudin et Param
par J. Chaleard

Ces résultats évoluent au fur et à mesure de vos votes. Néanmoins, **nous félicitons les auteurs!**

VOUS ETES DYNAMIQUE

Debug Magazine recherche pour son service publicité

ATTACHE (E) COMMERCIAL (E)

* Connaissance de l'industrie informatique souhaitée.
* Expérience des outils bureautiques (traitement de texte, bases de données, tableurs) nécessaire.

REMUNERATION MOTIVANTE EN FONCTION DES RESULTATS

Pour prendre rendez-vous, envoyez un Curriculum Vitae.

PETITES ANNONCES LECTEURS

Logiciels LOTO 6/49, sportif et TURF - LABO LOTO - EXPERT - GROUPAGES LOTO - FORMULA 2 - 1N2 SYSTEMS - SYSTEMA - QUINTESOFT - JP MACAIRE Fomboudeau - 33240 St ANDRE DE CUBZAC - Tél : 57.43.00.07. H.B. ou 57.43.61.33 après 19H.

Club Micro H.C.V. cherche News adhérents, fanzine gratuit.
1/73 rue Ile de France
59370 Mons en Baroeul

Vous avez des besoins en informatique. (Formation, conseil, développement d'applications...).
Alors, téléphonez au : (1) 30.38.15.04.

Des logiciels pour PC, professionnels, non protégés, intéressants et utiles à prix d'amis ? une seule adresse :CUMPS ASSOCIATION, BP 51, 40501 SAINT SEVER CEDEX (nouveaux tarifs, nouveaux logiciels).

Le serveur de l'achat/vente informatique par des particuliers, par des sociétés, neuf ou occasion, unité, disques, écrans, softs, etc... pour tout type d'ordinateurs.
Laissez votre annonce, c'est gratuit. 36 15 RSM*MATOS

Vends XT TANDY 1000SX. 640 Ko. DD. 20Mo. 2 lecteurs 5"1/4. Ecran CGA Couleurs + Mouse, Timer, Joystick et nbx logiciels originaux. Etat neuf, peu servi. Tél : 64.57.40.71.

Vends portable TOSHIBA 3100E ou portable TOSHIBA 1600. Disque dur. RAM 1Mo. Housse de transport et nombreux logiciels. 15 000F. pour le T3100E, 18 000F. pour le T 1600, à débattre. Se renseigner au : (1) 46.77.14.43

Pour les abonnés de DEBUG Magazine, les petites annonces sont gratuites (joindre la bande témoin du dernier numéro reçu). Pour les autres lecteurs, 50 f, par coupon renvoyé. Règlement par chèque ou carte bancaire à l'ordre de: DEBUG Magazine - 3, rue de l'Atlas - 75019 Paris.

J'ACHETE

J'ECHANGE

JE VENDS

DEBUG MAGAZINE

BULLETIN DE PETITES ANNONCES

☐ J'ÉCHANGE ☐ J'ACHÈTE ☐ JE VENDS

☐ Je suis abonné à DEBUG MAGAZINE : ci-joint bande témoin du dernier numéro

☐ Je ne suis pas abonné à DEBUG MAGAZINE; ci-joint un chèque de 50

DEBUG Magazine

MARCHE PC

LA PUBLICITE A DES PRIX ABORDABLES POUR TOUS !!!

Vous avez des produits ou services qui méritent d'être connus.
Vous voulez les promouvoir par des campagnes de publicité (achat ou vente).
Vous cherchez des rubriques spécifiques, claires, pratiques et faciles à trouver par vos prospects .
Vous désirez optimiser au maximum votre investissement.
Vous manifestez l'enthousiasme de réaliser des opérations suivies et répétées dans l'année.
Vous voulez un marché unique en France avec une transparence des prix incontestable.
Mais, vous ne disposez pas de budgets faramineux.

VOTRE SOLUTION : LES RUBRIQUES « MARCHE PC » !!!

1/2 PAGE : 3.950 F. HT

TARIFS 1991

1 PAGE : 7.900 F. HT
1/2 PAGE : 3.950 F. HT
1/4 PAGE : 1.980 F. HT
1/8 PAGE : 990 F. HT
1/16 PAGE : 490 F. HT

DEGRESSIFS

5 Parutions par an : 5 %
10 Parutions par an : 10 %
15 Parutions par an : 15 %
20 Parutions par an : 20 %

1/4PAGE : 1.980 F. HT

Marché PC de DEBUG Magazine, l'emplacement de choix pour trouver les produits référencés.

Renseignez-vous au 42.02.29.16

Demandez le Service Commercial

1/16 PAGE : 490 F. HT

CONDITIONS DE VENTE

1. **Couleurs du support + 20 %**
2. **Frais techniques en sus : forfait 500 F. HT**
3. **Paiement à la commande.**
4. **Insertion possible jusqu'à 10 jours avant parution.**

1/8 PAGE : 990 F. HT

SERVICE LECTEURS : 4

Entrez dans l'ère de la communication

Nous réalisons votre

SERVEUR MINITEL

Entrez dans l'ère de la communication
Nous réalisons votre
SERVEUR MINITEL
Logiciels pour créer un serveur minitel
Sur **RTC** ou **TRANSPAC** de 1 à 128 octets.

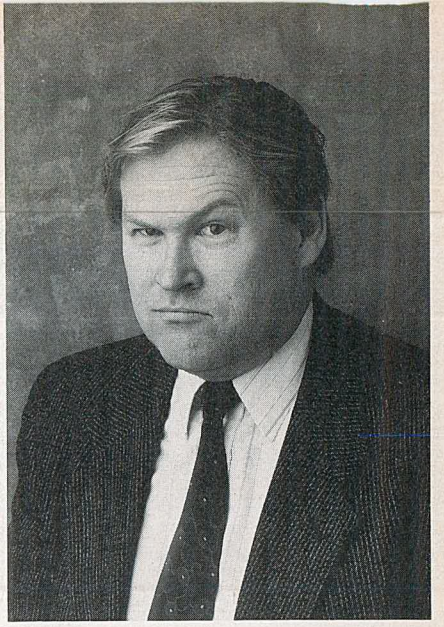
Pour **IBM** ou **Compatible PC**
2 voies RTC = 4 365 F. HT
4 voies RTC = 7 089 F. HT
128 voies TRANSPAC = 19 500 F. HT
Composeur dynamique vidéotex PRO
ECRAN VIDEOTEX = 3 365 F. HT

Pour **ATARI 1040** - 25 PRG = 2 990 F T.T.C
Serveur MUST monovoie + kit vidéotex

Ou nous vous hébergeons en **3614** ou **3615** sur nos ordinateurs à partir de 800 F. par mois.
Renseignements et documentation sur nos Serveurs minitel de démonstration
3615 IZEE ou **3615 TELEDIA**
Tél : (16) 33 53 13 66 / Fax : 33 04 50 30
Société MUST. BP. 4 - CARTERET 50270

John C. Dvorak

LA MAGIE : *CLEF DU SUCCES*



Si vous voulez gagner de l'argent dans l'industrie micro-informatique, attendez-vous à beaucoup de travail, dur et long. Malheureusement, le travail n'est pas tout, la magie a aussi son mot à dire. J'ai rencontré ce concept le jour ou j'ai parlé du succès de 1-2-3 avec un financier qui s'occupe des capitaux joint-venture. Je lui ai demandé ce qui , d'après lui, avait été à l'origine du phénoménal niveau de ventes, de l'acceptation et de la popularité de Lotus. « C'est de la magie », fut la réponse.

J'ai réfléchi sur ce problème et j'ai conclu que la magie doit effectivement faire partie de l'image. Certainement, 1-2-3 n'a pas été meilleur que Multiplan ou nombre d'autres tableurs. Et, pour expliquer le succès, il doit y avoir quelque chose de plus que l'organisation de la société Lotus. Beaucoup trop d'autres entreprises ont commencé avec de meilleurs plans de marketing, de meilleurs cadres, de meilleurs vendeurs et malgré cela, ont misérablement failli. D'autres entreprises, comme MicroPro, avec leur magique traitement de texte WordStar, ont prouvé que l'organisation de l'entreprise ne veut rien dire. C'est magique.

Les personnalités qui agissent dans l'ombre de l'entreprise ont l'air d'avoir quelque chose avec ce succès. Prenons, par exemple, Adam Osborne. Pendant son règne à Osborne Computers il a propulsé l'entreprise au plus haut niveau de la magie. Cette entreprise a grandi aussi vite qu'une entreprise peut le faire. L'influence de la magie a été intense. C'est évident, car son ordinateur portable était odieux, avec son minable écran de 5" qui ne pouvait afficher plus de 54 caractères sur une ligne. Sûrement, la magie a joué son rôle, elle seule peut expliquer sa phénoménale popularité. Lorsque la magie s'est dissipée, le toit s'est écroulé. Et cela devrait arriver, car il n'y avait rien pour le tenir. La magie a disparue quand Osborne et son épouse, fatigués, sont allés se reposer - trop tôt. Aucune fondation réelle n'a été bâtie et sans la magie rien n'a resisté.

Contrairement à la monté et à la chute d'Osborne Computers, l'entreprise récente d'Osborne, Paperback Software Company, est une organisation dépourvue de la magie, à ce que je peux voir. Je l'ai constaté avec stupéfaction suite à un article dans un de mes bulletins préférés, THE CPA MICRO REPORT (16800 W Greenfield Ave, Brookfield, WI 53005). L'auteur fait un commentaire suivant sur VP-Planner de Paperback Software : « Pouvez-vous croire cela, un programme meilleur que 1-2-3 et cela seulement pour 99.99 dollars? Nous ne l'avons pas cru tant que nous ne l'avons pas essayé ». Et il a continué à divaguer dans ce style.

Puisque il n'y a plus de la magie, seulement une analyse objective pouvait marcher. Le reste du monde est très dubitatif sur le produit et continue à pousser Lotus. Quelle peut être l'explication de ce phénomène? Quelque part, tout au long de son chemin, Osborne a perdu sa magie. Il ne sait certainement pas encore d'où elle est venue et comment elle est partie. Je ne le sais pas non plus.

La magie est si importante pour le succès que la majorité des entreprises qui vendent en détail la prennent en compte. Bien que peu de spécialistes de marketing utilisent le mot « magie », ils parlent de la « chose ». Est-ce que le produit a « quelque chose » ? Cette « grande chose ». Malheureusement, il est impossible de prédire quand on est tombé sur la « chose ». Vous pouvez essayer de le deviner ou commander les analyses du marché. Ces analyses peuvent aussi être trompeuses. Coca-Cola a expérimenté cela quand son département de marketing a déclaré que la nouvelle formule de Coke va être la préférée - elle a été sélectionné dans tous les tests. Mais elle n'avait pas la « chose ». La vielle Coke, elle, indéniablement beaucoup de la « chose ». Il est évident que c'est à cause d'un attrait magique que Coca-Cola a tellement de succès. Avouons le franchement : Coke ce n'est que de l'eau, sucrée et aromatisée. Pourtant, on la vend plus chére que la bière.

Revenons à nos moutons. Il n y a que le public qui peut dire si un produit est « magique » ou possède cette « quelque chose ». Certains produits, ont l'allure magique et étincelante et pourtant sont fades. D'autres produits, comme les hamburgers Big Mac, possèdent cette magie permanente et deviennent partie de la culture. Quand je vois le phénomène comme Big Mac, Lotus ou Apple, je me demande toujours quel était ce contrat obscur, conclu en échange du succès.

Selon moi, ce n'est pas une simple coïncidence que Steve Jobs ait passé beaucoup du temps dans les sous-bois des Indes, drapé d'un sheet, avant de faire sa fortune. Mitch Kapor est connu pour avoir enseigné la Méditation Transcendantale. L'épouse d'Adam Osborne, Barbara, s'occupe de la magie, a peu à voir avec sa nouvelle entreprise, mais elle a été le personnage clef dans Osborne Computers. Adam est né aux Indes. Tous ces personnages, ont-ils marché sous un trou d'ozone bien particulier ? Si oui, ils n'ont rien dit à personne.

Historiquement, beaucoup d' inventeurs et moguls, ceux qui sont devenus riches et célèbres, ont eu un passé mystique. Peut-être ont-ils trouvé une méthode pour induire à leur produits quelque chose qui crée cette attraction irrésistible.

Jetons un oeil sur quelque produits et idées magiques, pour voir ce que vous en pensez. Voici Apple, une machine dont le rapport qualité/prix est difficilement justifiable. Ils n'arrivent pas à en fabriquer assez ! Il y a aussi VAX de chez DEC. Vous auriez besoin de beaucoup de la chance pour vous mesurer avec. L'ordinateur CRAY est magique. Les oscilloscopes Tektronix. Tous ces produits ont pu passer par les périodes « magiques », ils sont devenus « réels » dans le sens qu'ils n'ont plus besoin de la magie. Ou en ont-ils encore besoin? Le mini-ordinateur HP 3000 semble avoir perdu beaucoup de sa magie. C'est parce qu'il est devenu trop vieux ? Il y a plein d'exemples de produits qui sont devenus trop vieux mais qui retiennent leur allure magique. La pétanque, par exemple.

Si le produit a assez de magie, il devient une partie permanente de la culture. Pour moi, il n'a rien de plus fascinant que le succès de la pizza délivrée à la maison. Cela ne vous frappe pas, comme l'exemple d'une affaire à succès ? C'est universellement accepté par tous les Pierres, Pauls et Jacques, dans le monde entier. C'est vraiment extraordinaire.

Mais, peut-être es-ce le concept en lui même qui est extraordinaire, qui sait ? La prochaine fois que vous achetez ou contemplez quelque chose, essayez de comprendre pourquoi vous le voulez réellement. Analysez et regardez si par hasard vous pouvez y détecter de la magie.

[SERVICE LECTEURS : INTERESSANT : 37/ MOYEN : 38 / MAUVAIS : 39]

COMPUMART

21, rue de Fécamp - 75012 Paris

Les prix discount !

LANGUE FORMAT PRODUIT	Prix HT	TTC
ACCESS SOFTWARE		
US 5 HEAVY METAL	239.43	284
US + WORLD CLASS GOLF SIM	274.95	326
ACCOLADE		
US 5 MEAN 18 GOLF	227.54	270
US 5 PINBALL WIZARD	211.50	251
US 5 SERVE AND VOLLEY	213.32	253
ACTION SOFT		
US 5 PERISCOPE	211.50	251
BLUE LION SOFTWARE		
US 5 TICKET TO PARIS	274.95	326
BRODERBUND		
US 5 ANCIENT ART OF WAR	227.54	270
US 5 SHUFFLEPUCK CAFE	211.50	251
US 5 WHERE IN EUROPE IS C	328.19	389
US 5 LODE RUNNER	237.02	281
DOLPHIN MARINE SYS		
US 5 DOLPHIN SAILING SIMU	284.43	337
ELECTRONIC ARTS		
US 5 SCRABBLE	274.95	326
US 5 STARFLIGHT	350.79	416
US 5 CHUCK JAEGER FLIGHT	274.95	326
US 5 GRANDSLAM BRIDGE	310.54	368
EPYX		
US 5 CALIFORNIA GAMES	201.51	239
US 5 SUB BATTLE SIMULATOR	274.95	326
US 5 WINTER GAMES	331.83	394
INFOCOM		
US 5 HITCHHIKER GUIDE GAL	227.54	270
US + BEYOND ZORK	78.18	93
US 5 ZORK TRILOGY	464.56	551
US 5 LEATHER GODDESSES	80.59	96
MICROCOMPUTEUR GAMES		
US 5 RAM	182.33	216
MICROPROSE		
US 5 GUNSHIP	296.31	351
US 5 CRUSADE IN EUROPE	303.39	360
US 5 F15 STRIKE EAGLE	213.32	253
US 5 SILENT SERVICE	213.32	253
US 5 DECISION IN DESERT	274.95	326
US 5 SOLO FLIGHT	274.95	326
US 5 CONFLICT IN VIETNAM	274.95	326
MINDSCAPE		
US 5 BALANCE OF POWER	251.24	298
NEW WORLD COMPUTER		
US 5 MIGHT & MAGIC II	331.83	394
ORIGIN SYSTEMS		
US 5 ULTIMA V WARRIORS DE	350.79	416
PARAGON SOFTWARE		
US 5 DR DOOM'S REVENGE	255.98	304
SIERRA		
US 5 KING QUEST	211.50	251
US 5 HELICOPTER SIMULATOR	211.50	US
5 KING QUEST IV	251.24	298
us 5 SPACE QUEST	211.50	251
US 5 LEISURE SUIT LARRY	201.51	239
us 5 SPACE QUEST 1 SARIEN	255.25	303
US 5 SILPHEED	184.88	219
SOFTWARE TOOLWORKS		
US 5 CHESSMASTER 2100	303.39	360
US + LIFE AND DEATH	303.39	360
SPECTRUM HOLOBYTE		
US 5 SOLITAIRE ROYALE	163.58	194
US 5 SOKO BAN	135.14	251
US 5 GATO	369.76	439
SPINNAKER		
US 5 SARGON III	251.24	298
SUBLOGIC		
US 5 JET SCENERY 3	109.39	130
US 5 JET SIMULATOR 2.1	251.24	298
US 5 JET SCENERY 11	111.44	132
US 5 JET SCENERY 7	109.39	130
us 5 JET SCENERY 1	109.39	130
us 5 JET SCENERY 2	109.39	130
TAITO		
US 5 SKY SHARK	237.02	281
THREE SIX SILICON		
US 5 DARK CASTLE	274.95	326

CONDITIONS DE VENTE :

1. Prix nets, frais de livraison inclus
2. Livraison dans la limite de stocks en 48 heures
3. Paiement seulement par chèque, carte de crédit (Visa, CB. MC) ou mandat
4. A cause de lois sur le copyright, pas de reprise de logiciels ouverts

Je veux recevoir la liste complète des prix.

Nom _____

Société _____

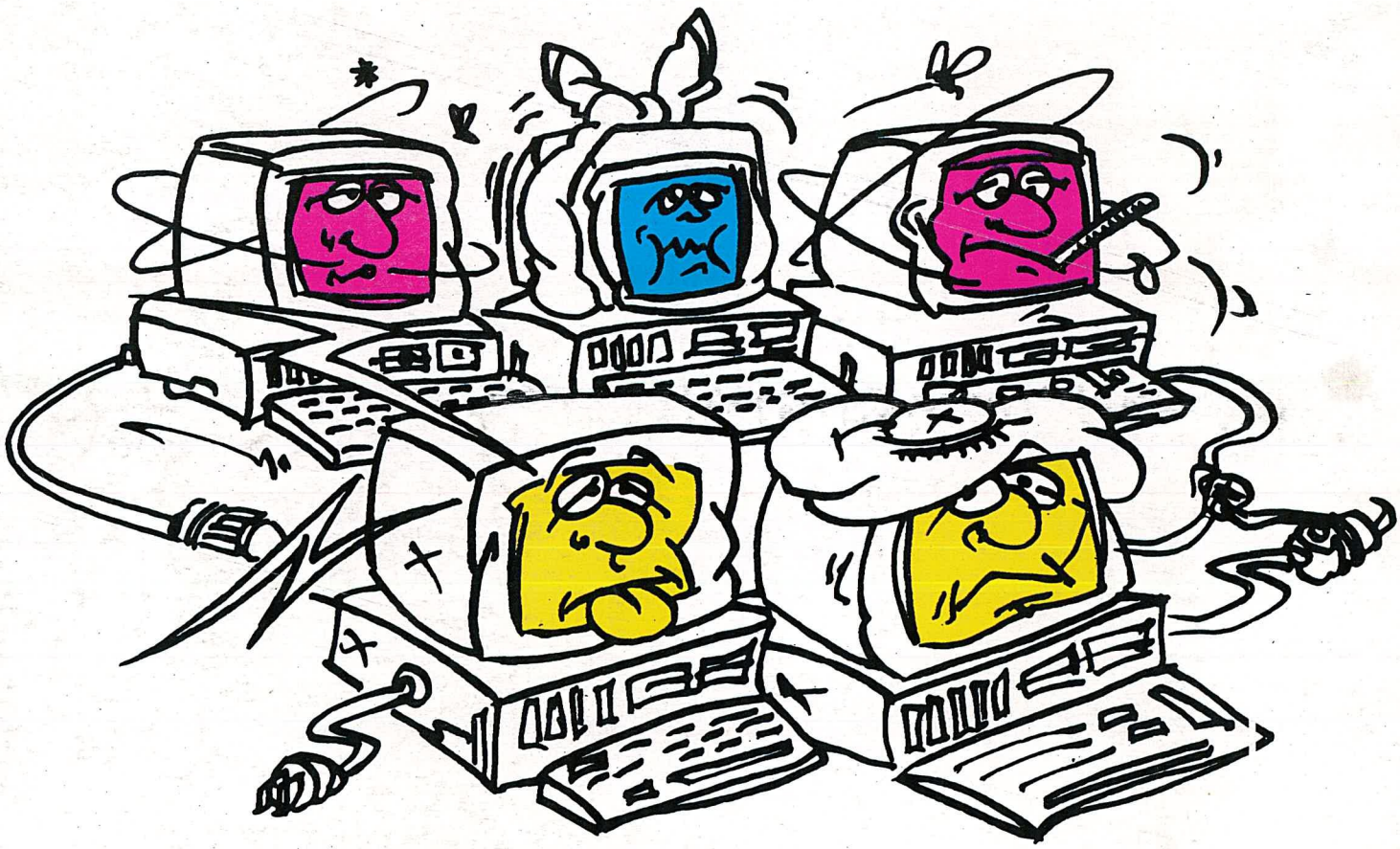
Adresse _____

Ville _____

Code Postal |_|_|_|_|_| Tél. : _____

BUGS? DEBUG!

DEBUG Magazine offre chaque mois les solutions techniques aux professionnels. Ils lisent DEBUG Magazine. Et vous?



abonnez-vous!